Volume 6, Issue 3 March, 1984
$1.50

TRS-80® Microcomputer News

■ LMP 2150 ■ PTC-64 ■ 8K PC-2 Program

Fort Worth Scene

I am writing this column on a warm (70 degrees) January day just after the coldest December in Texas. One result of that cold December is that all of our computers froze (figuratively, of course) which resulted in less material being produced for this issue.

LMP 2150

The LMP-2150 is Radio Shack's first Line Matrix Printer. Effectively, a line matrix printer prints an entire line at a time. This makes the printer much faster than other dot matrix printers which print a character at a time. See the LMP 2150 article for details.

With the introduction of a line matrix printer, we may need to clarify some terminology. Here at Radio Shack we have used the term Line Printer when we have talked about our printers. I think that usage began to distinguish printers which printed information one line after another from screen printers which printed information one computer screenful at a time.

In the printer industry, the printers which Radio Shack has always called Line Printers are known as Serial Printers. Serial because they print characters serially, one after another. A Line Printer prints a line at a time, like the new LMP 2150.

Confused? I hope not, but if you are, just remember that a rose is still a rose no matter what name someone puts on it. Radio Shack offers printers to fit every need and budget. Virtually any size, capability, or price of printer can be found at Radio Shack. If you are in the market for a printer, check Radio Shack's printers for price, performance and flexibility.

PTC-64

The PTC-64 is a printer controller (sometimes called a print spooler) which fits between your computer and your printer. The PTC-64 can't speed up your printer, but it can free your computer from having to wait for the printer during long print jobs.

I think the PTC-64 is one of the most compact and versatile printer controllers in the market today. The PTC-64 has its own Z-80 microprocessor, and gives you as much as 62K bytes of RAM for storing documents while they are being printed.

SHORT ARTICLES

We receive quite a few short articles every month. We use the ones that are immediately useful and file the rest for use at a later time as we need them.

As with many human endeavors, we sometimes lose track of this material. For this issue we have pulled some of the really good articles which just didn't quite fit from month to month.

We really appreciate the time and effort which many of you put into material that you send us. It is physically impossible to print all of the material, but we do want to acknowledge the valuable contribution that each person who sends us an article makes. Our thanks.

The LMP 2150 end PTC-64 are two new products in Radio Shack'a ever growing peripharel product iine.

TRS-80 Microcomputer News
Volume 6 Issue 3
MARCH 1984

TRS-80 Microcomputer News is published monthly by Redio Shack, e division of Tendy Corporation. A single six month subscription is available free to purchasers of new full size TRS-80 Microcomputer systems with addresses in the United States, Puerto Rico, Canada and APO or FPO eddresses. Certain smaller TRS-80 Microcomputers will not include this free subscription. Subscriptions to other addresses are not available.

The subscription rele for renewals and other interested persons with U.S., APO or FPO addresses is twelve dollers ($12.00) per yeer, check or money order. Single copies of the Microcomputer News may be purchased from Radio Shack Computer Centers or Computer Depertments for $1.50 suggested releil each.

The subscription rale for renewels and other interested persons with Canedian addresses is Fifteen dollers ($15.00) per year, check or money order in U.S. funds. All correspondence releled to subscriptions should be sent to: Microcomputer News, P.O. Box 2910, Fort Worth, Texes 76113-2910.

Retail Prices in this newsletter mey very at individual stores and deelers. The company cannot be liable for pictoriel and typographicel ineccuracies.

Back issues of Microcomputer News prior to January, 1981 are eveilable through your locel Radio Shack store as stock number 26-2115 (Suggested Releil Price $4.95 for the set). Back issues of 1981 copies are availeble as stock number 26-2240 (Suggested Releil Price $9.95 for the set). The 1982 back issues copies are availeble as stock number 26-2241 (Suggested Retail Price $12.95 for the set).

The TRS-80 Newsletter welcomes the receipt of computer progrems, or other meterial which you would like to make availeble to users of TRS-80 Microcomputer systems. In order for us to reprint your submission, you must specifically request thet your meteriel be considered for reprinting in the newsletter and provide no notice that you retain copyrights or other exclusive rights in the material. This assures that our readers may be permitted to recopy and use your meterial without creating any legel hessles

Material for publication should be submitted on magnetic media (tape, disk, or CompuServe). If you submit material on tepe or disk, and it is accepted for publication, we will send you two casselles or diskettes for each one you sent us. Cassettes will come from our box of mixed blank cesselles. If you submit material on CompuServe, and we think we mey use the materiet, we will extend your Microcompuler News subscription by six months for each article accepted. If you are submitting material over CompuServe, please include your neme end address or your subscription number so we can find you. If the material is very short, send it to us in E-Mail. If you have more than a few lines, you need to place the materiat in the ACCESS eree of CompuServe and then let us know it is there by leaving a messege on E-Mail.

Material mey be submitted by mail to P.O. Box 2910, Fort Worth, Texas 76113-2910, or through CompuServe. The Microcomputer News' CompuServe user ID number is 70007,535.

Programs published in the Microcomputer News are provided es is, for your information. While we meke reasonable efforts to ensure that the programs we publish here work es specified, Radio Shack can not assume eny liability for the accuracy either of the programs themselves or of the results provided by the programs.

Further, while Microcomputer News is e product of Radio Shack, the progrems end much of the informetion published here ere not Radio Shack products, end as such can not be supported by our Computer Customer Service group. If you heve questions about a program in the Microcomputer News, your first option is to write directly to the author of the program. When possible, we are now including author's addresses to facilitate communications. If the address is not published, or if you ere not heppy with the response you get, please write us here at Microcomputer News. We will try (given the limited size of our steff) to find en answer to your question and, in many cases, will publish the answer in an up-coming issue of Microcomputer News.

Trademark Credits

CompuServe™	CompuServe, Inc.
CP/M®	Digital Research
Dow Jones™ NEWS/RETRIEVAL Service®	Dow Jones & Co., Inc.
LDOS™	Logical Systems, Inc.
VisiCalc®	VisiCorp, Inc.
XENIX™	Microsoft
Program Pak™	Tandy Corporation
SCRIPSIT™	Tandy Corporation
TRSDOS™	Tandy Corporation
TRS-80®	Tandy Corporation

TRS-80® Microcomputer News

Contents:

Prices shown in TRS-80 MICROCOMPUTER NEWS are in U.S. Funds.

Two Classroom Packages by Children's Television Workshop Now Available From Radio Shack

(Editor's Note: Children's Computer Workshop, an activity of Children's Television Workshop, developed the packages described in this article. Because the Children's Computer Workshop is a division of Children's Television Workshop, the Children's Television Workshop name was used in this article.)

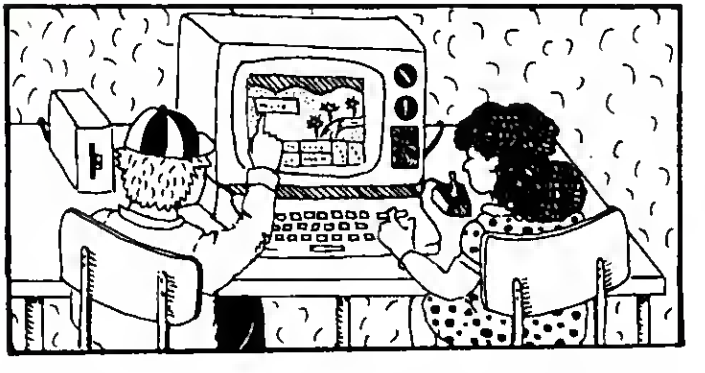
Two classroom packages developed by the Children's Television Workshop now are available through your local Radio Shack store or Computer Center. HANDS ON! is an introductory computer literacy package. PLAY WITH LANGUAGE is a Language Arts package for beginning readers. Both packages are designed for use on a 32K Color Computer disk system with Disk Extended Color BASIC. A joystick is needed to use PLAY WITH LANGUAGE.

HANDS ON! contains two computer-literacy learning modules: BLACKBOARD and COLOR IT. BLACKBOARD introduces students to some of the communication capabilities of a computer. The program is a beginning word-processor that allows students to write on the "blackboard," then edit their work. Students can also set up a personal file system and edit it as desired. Finally, the activity provides the beginning of an electronic mail system in which students share their writing with others. Through BLACKBOARD, students are encouraged to explore the computer's capabilities as a tool for electronic manipulation of words and information.

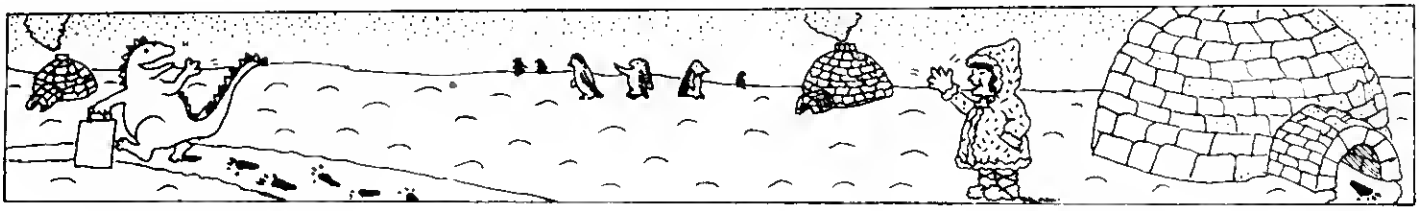
COLOR IT, the second computer-literacy program in HANDS ON!, demonstrates computer graphics by allowing students to draw pictures using the up, down, left and right arrow keys on the Color Computer. Students can control the color, shape, and speed of the cursor. The (P) key is used to switch between pen down (drawing) or pen up (moving the cursor without drawing). The (F) key allows students to use computer functions such as shuffling colors, flipping a picture, or moving a picture pixel by pixel.

PLAY WITH LANGUAGE contains three language arts modules: PICTURE PLACE!, ROLL-A-WORD, and BAGASAURUS.

In PICTURE PLACE! students associate sight words or beginning vocabulary words and their picture referents. The student chooses a background scene and then creates a picture using words as building blocks. The student selects words to move across the screen. When the student decides that a word is in place, he or she can change the word into a picture.



ROLL-A-WORD is a linguistic activity which requires students to match pictures and words in order to complete sentences in original and clever ways. Words and their picture referents are available in rhyming patterns, which means that increasingly successful play is linked to mastery of rhyming.



In BAGASAURUS, students practice reading comprehension skills such as classification, following directions, and sequencing. Students also develop vocabulary through the use of synonyms, antonyms, and multiple meanings of words. The student is on a journey with a creature called BAGASAURUS. The student's task is to collect words and images by correctly answering questions. These words and images are used in stories later in the program.

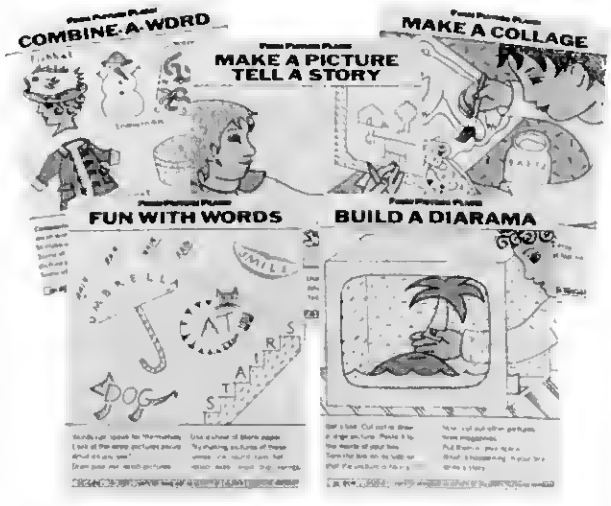
CLASSROOM MATERIALS IN HANDS ON! AND PLAY WITH LANGUAGE

HANDS ON! and PLAY WITH LANGUAGE contain not only computer programs, but also a variety of ancillary teaching materials, and special features in the software designed to enhance the teaching experience.

A Learning Manager feature in the computer programs for BLACKBOARD, COLOR IT, PICTURE PLACE!, and ROLL-A-WORD allows the teacher to assign students to different levels of play, to customize lessons, or to save a student's work on disk.

Teaching Extensions in the HANDS ON! and PLAY WITH LANGUAGE manuals provide suggestions for cognitive and affective activities keyed to the content and educational goals of each computer activity.

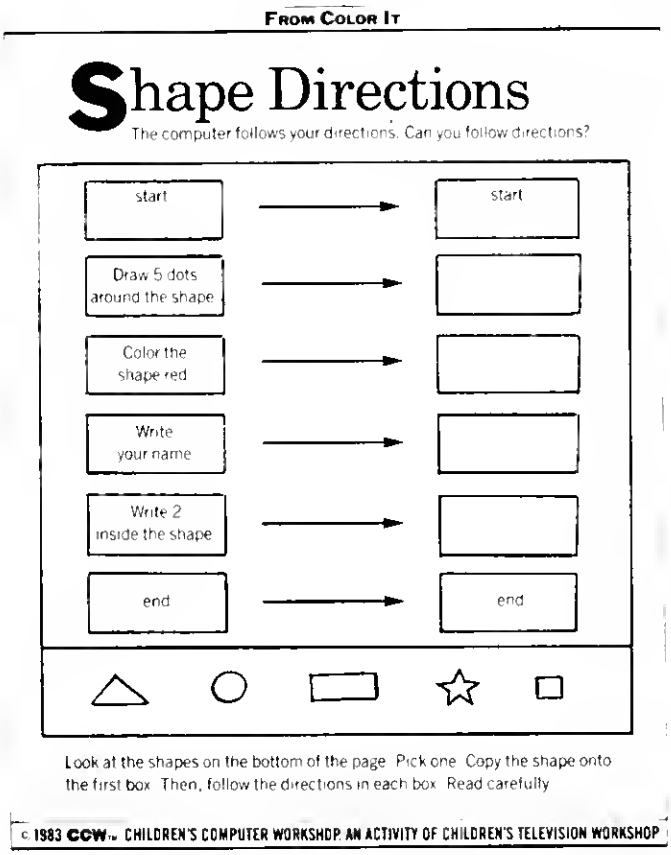
Activity Cards are keyed to each computer activity. These bright, laminated cards contain ideas and instructions for more non-computer activities.



A colorful Game Board for each activity allows students to play while reinforcing their understanding of the instructional content.

Worksheet masters are included, to be reproduced by the teacher for use as class handouts. These also focus on the same educational goals as the software.

FROM COLOR IT



HOW YOU CAN OBTAIN THESE PROGRAMS

HANDS ON! (Cat. No. 26-2539) and PLAY WITH LANGUAGE (Cat. No. 26-2538) are now available from Radio Shack. Your local Radio Shack Computer Center may have these packages in stock, or they can be ordered through your local Radio Shack store or Computer Center. Suggested retail price for each package is $99.00 (prices may vary at individual stores and dealers).

Two groups of educational games developed by Children's Television Workshop are available also from Radio Shack. Basic skills games for ages 3-6 feature the Muppet stars of Sesame Street. Included are Grover's Number Rover, Big Bird's Special Delivery, Ernie's Magic Shapes, and Cookie Monster's Letter Crunch. Games in the "cooperation and strategy" series for ages 7 and over include Star Trap, Peanut Butter Panic!, and Taxi. The games run on a 16K Color Computer cassette system with Extended Color BASIC and joysticks. For more information, contact your local Radio Shack store or Computer Center.

The LMP 2150, A Line Matrix Printer

By Bruce Elliott

The LMP 2150 is Radio Shack's first Line Matrix Printer (LMP). Built to take the pressure of heavy-duty processing, the LMP 2150 is superb for big jobs like database reports, inventory reports or payroll.

WHAT IS A LINE MATRIX PRINTER?

Up to this point in time, Radio Shack has offered two basic types of printers: Serial Dot Matrix and Letter Quality.

A Serial Dot Matrix printer from Radio Shack can be described as a printer which uses columns of dots to produce readable characters. The dot columns are a single character in height, and they are placed on the paper one column at a time by a print head which moves back and forth across the full width of the line being printed. Radio Shack has used a number of technologies to form the dots, including Impact, Thermal, and Ink Jet.

A Letter Quality printer from Radio Shack is a printer which produces fully formed characters by using a Daisy Wheel.

Now we come to Line Matrix Printers. A line matrix printer is a printer which prints ALL of the characters in a given row at one time. Gone is the print head which travels back and forth across the entire print line to create the characters. The primary advantage of a Line Matrix Printer is speed.

The LMP 2150 is recommended for the printer user who needs a heavy duty, long life printer requiring minimal maintenance to keep it printing. The high speed of the LMP 2150 (80 to 150 lines per minute), combined with its large character set and multiple printing fonts means that you don't have to buy several printers to get your work printed.

The superior print quality of the LMP 2150 is produced using patented HammerBank technology. The Hammer Bank consists of 17 heavy duty, stored energy spring leaves. Each of these leaves prints the dots for eight characters in each line. The crisp, uniform dots overlap to create characters so solid they are clear to the last copy of a six-part form.

Quality construction and design give the LMP 2150 the highest mean time between failure (MTBF) rates in the industry. What this means to you is that you can expect the LMP 2150 to print heavy workloads with a minimum amount of preventative maintenance. Periodic readjustments typical of other printers are not needed for the LMP 2150. The print mechanism's one-piece unitized assembly gives precision dot placement that won't shift or get out of register. The only maintenence needed is periodic cleaning, typically every six months or 500 hours of operation.

LMP 2150 CHARACTERISTICS

Data processing characters (normal, compressed, and condensed) are created using a 5×9 dot matrix pattern. Correspondence quality characters are created using a 7×12 dot matrix pattern.

For maximum efficiency, the LMP 2150 operates in one of three modes:

- Data processing mode for fast output
- Word processing mode for letter writing
- Graphics mode for drawing pictures, or graphs.

The LMP 2150 can handle continuous fanfold paper ranging in width from 3-inches to 16-inches including the guide hole side strips. The printer can print one original with up to five copies (with or without carbon). Paper stock weights can range from 15 to 100 pounds.

LMP 2150 comparative lines per minute (LPM) print speeds (no descender characters) among its various character sets:

Correspondence Quality—10 CPI	80 LPM
Condensed—16.7 CPI	100 LPM
Standard—10 CPI	150 LPM
Compressed—12.5 CPI	125 LPM

Characters per line:

Correspondence Quality—10 CPI	132
Condensed—16.7 CPI	220
Standard—10 CPI	132
Compressed—12.5 CPI	165

Dot positions per Horizontal line -

Correspondence Quality - 10 CPI	2640
Condensed - 16.7 CPI	2640
Standard - 10 CPI	1584
Compressed - 12.5 CPI	1980

In the Graphics Mode there are 7 vertical dots available in each dot column and a variable number of horizontal columns depending on the mode the printer is in:

Condensed	1320
Standard	792
Compressed	990

On a per inch basis, all three graphic modes print 72 dots per inch vertically. The Condensed mode yields 100 dots per inch horizontally, while the Standard mode gives 60 dots per inch and the Compressed mode gives 75 dots per inch horizontally.

Character Sets (Number of characters in each set)

ASCII	94
American Special	32
European WP	32
Block Graphics	30

The LMP 2150 uses a built-in 8-bit parallel interface to the computer.

The physical size of the printer is:

24.6 inches wide
10.3 inches high
20.7 inches deep
60 pounds

The LMP 2150 (26-1272 $3995) is available at more than 1000 Radio Shack Computer Centers and participating Radio Shack Dealers nationwide. The 1-inch wide nylon fabric ribbon (26-1287, $12.95) is capable of printing 15 million characters per ribbon. The LMP 2150 is backed by dependable Radio Shack service, with on-site service contracts available.

The matching printer stand (26-4307 $149.95) is recommended for use with this printer. Regular printer stands may not hold up well under the stresses imposed by the LMP 2150.

LRL Variables

Don Hallden
4629 W Walnut #2033
Garland, TX 75042

I have found the VAL function in Model III Disk BASIC to be of use in connection with defining FIELD lengths when using Random Access Files. The FIELD statement normally requires that the lengths of each field be entered as constants. If it is desirable to use variables for field lengths, this can be accomplished by entering the field length as a character variable and then applying the VAL function to the string variable in the FIELD statement.

Example:

```
100 LINE INPUT "ENTER LOGICAL RECORD LENGTH: "; LRL$
110 OPEN "R", 1, "TESTFILE", VAL(LRL$)
120 FIELD 1, VAL(LRL$) AS A$
```

Note that the following will give a syntax error when RUN on the Model III:

```
100 INPUT "ENTER LOGICAL RECORD LENGTH: "; LRL
110 OPEN "R", 1, "TESTFILE", LRL
120 FIELD 1, LRL AS A$
```

Defining Files for Profile Multi-User

The Small Computer Company
P.O. Box 2910
Fort Worth, TX 76113-2910
By Ivan Sygoda, Director, Pentacle

This month we'll take our first close-up look at the multi-user version of Profile (26-6412, $499). It runs under TRS-XENIX on the Model 16 series of computers, and it's packed with features that give it power and flexibility only available, until now, on much more expensive machines.

Although Profile 16 is very sophisticated, from the user's point of view, it is as simple—or even simpler—than Profile Plus or Profile III Plus. It is based on the same modular principles that have made the earlier Profiles so successful. I'll illustrate by setting up a relatively simple data base called "concerts," which I use to keep track of the dance concerts I'm supposed to attend as part of my business at Pentacle.

Number key segment:	Field Heading	Len	Type
1	Pentacle member, Audit (*/@)	1	*
2	Artistic Director last name	15	*
3	Artistic Director first name	12	*
4	Company Name	30	*
5	Company alpha key	10	allup
6	Concert location	15	*
7	Phone no. for reservations	8	phone
8	reservation made? (y/n)	1	yesno
9	A1) Day	3	day
10	A2) Date	8	mdy/
11	A3) Time	5	hrmin
12	A4) Go?(+)	1	*
13	A1) Day 2	3	day
14	A2) Date 2	8	mdy/
15	A3) Time 2	5	hrmin
16	A4) Go 2? (+)	1	*
17	A1) Day 3	3	day
18	A2) Date 3	8	mdy/
19	A3) Time 3	5	hrmin
20	A4) Go 3? (+)	1	*
21	A1) Day 4	3	day
22	A2) Date 4	8	mdy/
23	A3) Time 4	5	hrmin
24	A4) Go 4? (+)	1	*
25	A1) Day 5	3	day
26	A2) Date 5	8	mdy/
27	A3) Time 5	5	hrmin
28	A4) Go 5? (+)	1	*
29	Note	70	*

Key segment record length: 247
There is no data segment.

Figure 1: The Fields Defined for "concerts"

Along the way, I'll concentrate on the ways Profile 16 differs from its TRSDOS-based cousins.

IT'S A SCREEN!

Figure 1 lists the fields I defined for "concerts." The illustration resembles the print-outs you get with Profile II and III, but if you look closely, you'll see some significant differences—the labels for associated fields and the strange names listed under "type." I'll come back to these in a moment. One of the most wonderful enhancements can only be seen on the "Define Files" screen itself. Instead of viewing fields one by one, they all appear on the screen as you create them. You can shuffle the fields around, change their lengths and headings and then shuffle them again until you're satisfied with the result. In addition to being faster and more convenient, it lets you keep the total picture in mind as you go along.

Associated fields have been improved tenfold. Profile Plus and Profile III Plus allow you to tie the fields together by grouping them under any of the 26 letters of the alphabet. This establishes a vertical list, so to speak, through which the computer can scan for a particular item. Profile 16 lets you enlarge this list into a two-dimensional grid by specifying letter-number combinations.

Figure 2 illustrates how I applied this to "concerts." When dance companies are invited to perform, they are generally expected to give three to five performances spread out over a weekend. I've numbered the performances 1 to 5. The first column lists the day, which is associated field A1 (see figure 1 again). The second column shows the corresponding date, associated field A4, where I indicate which performance I plan to attend. I can scan, sort, and select according to any item in one of the vertical columns.

For instance: Which concerts can I attend on Friday? Is there a matinee on the 12th? When am I supposed to see the Acme company? I simply print a chronological list of next week's performances.

This "grid" can be up to 26 rows deep (A-Z) and up to 10 columns wide (A0, which is the same as A, to A9, B0-B9, etc.). What's more, the "horizontal" information tied to a particular item always surfaces with it! That's rather amazing.

Associated fields can be tricky. (See my article in the April 1983 *Microcomputer News*.) Profile 16 has certain functions that enable you to work your way around the pitfalls. We'll examine these in coming months.

TYPE CASTING

Look again at figure 1. About the only thing you'll recognize (from "Define Screens" in the earlier programs) in the list

```
Dec  1 13:46 1983    File Name: concerts              Screen #1    Page 1

Concert Schedule
Name of Company::::: *4                                / Alpha key: *5        /
Artistic Director::: (Last) *2                 / (First) *3          /
Type <*> if Pentacle member, <@> if concert is to be audited: *1

Location of Concert: *6              /
Phone number for reservations: *7      /      Reservation made? (y/n) *8

          Performance    Day         Date          Time       Co?(+)
               1         *9 /      *10      /      *11  /       *12
               2         *13       *14             *15         *16
               3         *17       *18             *19         *20
               4         *21       *22             *23         *24
               5         *25       *26             *27         *28

Note:  *29                                                                  /
.....................................................................................
Record number !@RN    /
Created !@CD     by !@CB    /
Updated !@UD     by !@UB    /
Last batch update:  !@BD    /
```

```
Cursor path:

    4       23
    5       25
    2       26
    3       27
  TAB       TAB
    1       29
    6       TAB
    7       12
  TAB       16
    9       20
   10       24
   11       28
   13       TAB
   14        8
   15
   17
   18
   19
   21
   22
```

Figure 2: Screen 1 Format and Cursor Path

of field types is "*". This is the default type: letters, numbers, punctuation, and spaces. In Profile II and III, field types were defined when you set up your screens. In Profile 16, they are specified at file definition. This gives you access to a whole range of field edits that have two main purposes. First, they make data entry more accurate by permitting the user to enter only the type of information that is appropriate to a particular field. Second, they make data entry faster and easier because you can use them to create short-cuts—automatically formatting an entry or supplying characters, for instance.

The alphanumeric ("*") edit type is one of many system edits that govern entry of numbers, dates, and times. The "." edit allows digits and the decimal point, automatically supplying two decimal places and right-justifying the number in the field. ".n" works just like the preceding but supplies 0 to 8 decimal places, depending on what number replaces the "n". "F" indicates floating decimal format.

There are 12 kinds of date formats allowing almost any permutation; from slashed to unslashed, and from two- to four-digit years. More important, every one of them sorts properly! Entering "/" into a date field automatically supplies today's date in the requested format. Shortened entries involving slashes automatically "fill-out." For instance, if you type "4/1/84," it appears on the screen as "04/01/84." No fooling!

Finally, there are two time edits: HMS for hours:minutes:seconds, and TIME, which accepts 00:00:00 to 23:59:59. Entering a colon will supply the current time, and abbreviated entries will be expanded just as date entries are. The White Rabbit would have loved it. As you can see, I use the "MDY/" date edit for fields 10, 14, 18, 22, and 26.

System edits are built into the system and cannot be changed; they are simply there for you to take advantage of as you wish.

GLOBAL CONCERNS

Another group of edits, called global edits, also comes with Profile 16. Like the system edits, global edits are built into the system and can be used in any data base file that calls them; however, unlike system edits, they can be modified and added to by the user. The limit is 100.

The global edits that change the format of entries in various ways are the ones you'll probably use most. For example, the "$" field type supplies the dollar sign, accepts digits from 0-9, adds the point and two decimal places, and right-justifies the result. "PHONE" accomplishes one of two things, depending on the field length and the number of characters entered. It automatically turns 8173903935 into (817) 390-3935 and 5551212 into 555-1212. "SSNUM" turns 123456789 into 123-45-6789. "ZIP" accepts either 5 or 9 digits, automatically supplying the dash for the ten-character code.

Another type of global edit restricts data entry to sets of alternatives. "YESNO" lets you enter only "Y", "y", "N", or "n" and converts the character entered to upper case. "SEX" does the same thing for "F" or "M".

A third group of global edits makes cosmetic changes in input. "ALLUP" converts all characters entered to upper-case. I used this edit in my field 5 to ensure that my alphabetization key would not be affected by upper/lower case differences. "UPLOW" converts only the first character of a word, usually a name, to upper-case, and the following character to lower-case.

The remaining global edits "filter" data entry, and are used mainly to build other edits, both global edits and local edits. Local edits are entirely user-defined. Whereas you can have up to 100 global edits in your entire data base, you can define up to 100 local edits for each data base file. Local edits apply only to the file in which they were created.

```
--Name-- ------------------------------ Edit ---------------------------------

HRMIN     \ N N <":"> N N | N <":"> N N
DAY       ^_"th"<u>|"su"<n>|"m"<o><n>|"t"<u><e>|"w"<e><d>|"f"<r><i>|"s"<a><t>
```

Figure 3: The Edit Dictionary for "Concerts"

LOCALS

I made two local edits for "concerts." Since both time edits count seconds, which are irrelevant for concert starting times, I invented "HRMIN." "830" becomes "8:30" automatically. "DAY" changes "M" or "m" to "mon," "T" or "t" or "tu" to "tue," etc.

Figure 3 shows the local "Edit Dictionary" for "concerts." These edits could have just as easily been added to the list of global edits and thereby be available for all files. The letters and punctuation marks are all part of the syntax used to define an edit. They form a concise and powerful "language," which we'll explore in a future article.

FURTHER AFIELD

Looking again at figure 2, you'll notice something else new. The two-letter combinations preceded by "@" are "system-maintained" fields. They automatically keep track of such things as record number and date of last update without you having to define fields for this information or enter it manually. The system "remembers" it whether or not you choose to display it on a screen or report. The complete list of system fields is shown in figure 4, with the length allotted to each.

System-Maintained Fields

@RN	record number; leave eight spaces
@TD	today's date, MM/DD/YY; leave eight spaces
@CD	date record first used; leave eight spaces
@UD	date last updated; leave eight spaces
@BD	date last updated in batch; leave eight spaces
@ID	operator ID; leave eight spaces
@CB	record first used by; leave eight spaces
@UB	last updated by; leave eight spaces
@TM	time; leave eight spaces
@SN	screen number (processing only)
@PN	page number; leave five spaces
@RS	number of records selected per subtotal section; leave eight spaces
	tents of subtotal field
@SH	subtotal field heading
@DT	date, spelled out (i.e., Feb. 10, 1985); leave 15 spaces
@FN	format name
@TS	total number of records selected

Figure 4: System Fields

HEY, DUMMY!

There's yet another type of field which is available for display on screens and in reports, but which isn't defined when you set up your file. This is the so-called "dummy" field, which is defined on the processing tables you can create for your data base. I don't have space this month to describe the processing tables, but here's a simple example to whet your appetite.

My field 2 is a last-name field (length 15) and field 3 is a first-name (length 12). There is no "push lett" field for screen formats, but I can achieve the same result by using the automatic processing table, as follows. Dummy fields are assigned names consisting of one or two letters. Since what I want to appear in this field is first name/space/last name, I write the following "formula" on the first "action line" of the table: NM=32. I allow 28 spaces on the screen, since that is the maximum possible combined length of the dummy field. Whenever the record is accessed by Profile, the name appears on the screen where indicated by "!NM". As you'll see in future articles, the processing tables are very talented.

FOR PROFILE PLUS AND III PLUS USERS: A BETTER WAY TO MAKE BACKUPS

Every manual you've ever read cautions you to make frequent backups of your data diskettes. Some users follow this advice and never have a problem. Others take chances and eventually learn the hard way—by losing hours or even months of work.

Diskettes are fragile, and any of a dozen environmental hazards can make your data unreadable. That's reason enough to become religious about backup procedures. Yet we recently had a misadventure at Pentacle against which our normal routine was insufficient protection. We have a three-drive Model III, which is often used for three-disk Profile files (one runtime disk, write-protected, with all the format programs, and two data disks, which we back up after each work session).

To make a long story short, one of the drives went out alignment, but slowly. It worked until the moment it crossed some mysterious threshold, whereupon it developed a taste for destroying disks.

Of course, we got the computer fixed immediately. But then the backup disks wouldn't work! Since they had been made on the misaligned drive, they could only be read by an equally misaligned drive. Rivulets of sweat coursed down my body as I pictured an irate client and twelve months of work out the window.

I put the disks in my briefcase and made the rounds of everyone I knew with a Model III, seeking a needle in a haystack—a drive not so misaligned it wouldn't work, but enough to read my orphaned disks. Believe it or not, I found one, and saved both my data and my derriere.

THE MORAL

Never make backups in such a way that all copies of a disk are created in any one drive. If you backup from :1 to :2 to update copy A, switch the source disk and go from :2 to :1 to make copy B. In other words, don't put all your eggs in one drive.

PROFILE Editor's Note: This is Mr. Sygoda's sixteenth article in a series of 'how-to' Profile articles. We hope that you enjoy this feature, and we look forward to your comments and questions on Profile.

Pentacle is a New York City-based non-profit service organization specializing in administrative services for performing art groups.

Xenix cu Utility

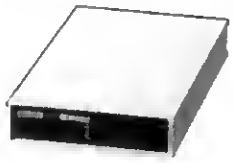
Call up is a communications program included free of charge as a utility on the Xenix 1.3 core diskette. Version 1.3 of the core package should be readily available at the time of this writing. If you are running an older version, please visit your nearest Radio Shack Computer Center and ask them to order the following upgrades for you. If you have never initialized the hard disk with Xenix, order 700-2052. If you are upgrading your system from a previous version of Xenix, order the upgrade 700-2066. This article assumes you are working with Xenix version 1.3

The cu utility allows one Xenix system to communicate and exchange files with other Xenix systems. It is also possible to use cu to dial some non-Xenix systems. Please notice I said some non-Xenix systems. Cu uses an asynchronous protocol and an ASCII character encoding scheme. The default UART settings are an eight bit word length, no parity bit, and one stop bit. The user can temporarily modify the word length and the parity, but the communication protocol, the number of stop bits, and the ASCII character encoding are fixed. So if you need to transmit a file to a system using two stop bits or a synchronous protocol, cu should not be used.

The first step in readying your system to use cu is to disable device tty01. To do this type:

disable tty01 (ENTER)

The Radio Shack modem you purchase for use with cu should be a Modem II (26-1173) or a DC-1200 baud modem with auto dialer board (76-1005 & 76-1009). These two modems possess auto-dial and auto-answer capabilities. If you choose to use any other modem, be aware that the call up utility's auto dial features will not work properly.



Read the addendums provided with the Xenix 1.3 core software to determine the correct method of configuring your modem. Next connect one end of an RS232C cable to the top most DB-25 connector at the rear of the computer. The other end of the cable should then be connected to the DB-25 found at the rear of the modem.

In our first example let us assume you are going to manually dial a number using cu. To start the cu utility you would type the following:

cu wait –s baud rate –a /dev/null –l /dev/tty01 (ENTER)

The message "connected" should appear on the screen. At this time, manually dial the phone number and place the modem in the originate mode. You are now on line with the host system.

To disconnect from the host and exit the cu program, type:

(CTRL) **6**

You should now be back at the shell prompt.

Now let us use the auto-dial feature of cu to call up the same system. Type:

cu phone number –s baud rate

That is all there is to auto dialing. When the carrier is received by your modem, the message "connected" will appear on your screen. You are now on-line with the host.

Now that we have dialed the system and logged in, let us examine the uploading capabilities of cu. If you are dialing another xenix system, you can simply use the "put" command as shown in the manual. Again press (CTRL) 6. You should see the tilde symbol appear. Then continue typing on the same line:

%put from source-filename to destination-filename

This command takes the source-file from the local system and copies it to the destination-filename on the remote system. This command requires that the directory you are writing to have write permission for the category "other". If you are unfamiliar with changing directory permissions, read up on the chmod command in your manual.

If you are connected with a non-xenix system, you should type:

(CTRL) **6 < filename** (ENTER)

This command dumps the contents of the file from the local system out to the RS232C port. The software on the remote system must be capable of capturing the information into RAM or writing it to disk using xon/xoff flow control.

The download features of cu seem to confuse many of its users. When downloading to another xenix system, you can use the "take" command. This command takes a file off the remote system and transfers it to the local system. The command "take" is shown below.

(CTRL) **6%take from source-filename to destination-filename**

The process becomes a bit more difficult when connecting with a non-xenix system.

All the information required to download a file from a non-xenix system to a xenix system is provided by the non-xenix system. The sender must transmit the following to the xenix system.

CTRL J
CTRL 6 destination-filename CTRL J ENTER

This command line starts the file diversion into the file specified on the local system. Again let me stress that nothing is typed from the xenix system. All these commands are sent from the remote location to you. Any data following the command line will be routed to the file specified in the command line. Following the data should be the command line to end the diversion and close the file. That command is:

CTRL J
CTRL 6 > CTRL J ENTER

The destination-filename used by the remote system must reside in a directory which grants write permission to the "other" category.

The last command of cu I wish to cover is the command that temporarily exits cu and invokes the interactive shell of the local system. It is while using this command that the user can change the default word length and parity settings. While in cu type:

CTRL 6!

This command jumps back to the shell prompt where the user can then use the "stty" command to change the word length and parity parameters. To return to cu, simply press:

CTRL d

I hope this brief introduction to the call up utility will spark your curiosity in Xenix communications. The utility is relatively easy to use once you get the hang of it.

Typewriter

Jay A. Jolicoeur
Cullinan Engineering Co., Inc.
200 Auburn St.
Auburn, MA 01501

Below is a modification to David Salisbury's typewriter program. I use this program with my office Model II.

My main problem is that I make a lot of typing mistakes and found myself retyping quite a few letters, but they weren't long enough or important enough for SCRIPSIT.

This program is a modification of Mr. Salisbury's program using an input routine that I use on most of my BASIC programs. It allows the user to type in the entire line before it is printed on the printer and make any necessary corrections. This saves a lot of correction fluid. Also, a blank, reverse line the length of the designated line is displayed to help determine when the margin is near.

With this program I no longer have any need for my typewriter or messy corrections. Now, all I need is to create a spell check program.

```
1ØØ '* * TYPEWRITER * *
11Ø CLEAR 1ØØØ
     : DEFINT A-Z
12Ø CLS
     : PRINT @ (1, 25), "* * Model II Typewriter *
     *";
     : PRINT @ (2, 3Ø), "Press [F1] to exit";
     : PRINT @ (3, 25), "Set margins (Ø - 132) Left
     ";
     : LINE INPUT L$
     : L=VAL(L$)
     : PRINT @ (4, 46), "Right  ";
     : LINE INPUT A$
     : A=VAL(A$)
     : R=A - L
     : CLS
13Ø PRINT TAB(L) CHR$(26); STRING$(R, " "); CHR$(25);
     : PRINT @ (ROW(X), L), "";
14Ø F=Ø
     : W=1
     : W$=STRING$(132, " ")
15Ø C$=INKEY$
     : IF C$="" THEN 15Ø ELSE IF C$=CHR$(1) THEN
     LPRINT CHR$(12)
     : END
16Ø IF C$=CHR$(8) THEN PRINT C$; " "; CHR$(28);
     : W=W - 1
     : MID$(W$, W, 1)=CHR$(32)
     : GOTO 15Ø
17Ø IF POS(X)>A THEN F=1
18Ø IF F=1 THEN IF C$=CHR$(32) THEN C$=CHR$(13)
19Ø MID$(W$, W, 1)=C$
     : W=W + 1
     : PRINT C$;
2ØØ IF C$=CHR$(13) THEN LPRINT TAB(L) MID$(W$, 1,
     W-1);
     : GOTO 13Ø ELSE 15Ø
```

Computer Customer Service Address and Phone Numbers

8AM to 5PM Central Time
Computer Customer Services
400 Atrium, One Tandy Center
Fort Worth, Texas 76102

Productivity/Special Applications	(817)338-2390
Accounting Software	(817)338-2391
O/S and Languages, Group No. 1	(817)338-2392
O/S and Languages, Group No. 2	(817)338-2393
Hardware and Communications	(817)338-2394
Home Software	(817)338-2395
Educational Software	(817)338-2396
Newsletter Subscription Problems	(817)870-0407

XENIX Notes

Kraig Snodgrass
Drake Univarsity Physical Plant
1422 27th
Des Moines, Iowa 50311

I am writing in response to your October 1983 *Microcomputer News* article about the TRS-XENIX operating system. The article was titled "TRS-XENIX POWER" and appeared on pages 24-27. Information contained within the article has been very helpful and timely in the startup of TRS-XENIX here at Drake University.

After following your instructions on pages 24-25 in reference to the user shutdown procedure, I found what appeared to be a major problem with the method used to perform a shutdown. When the command for shutdown is put into a .profile file in the /usr/shutdown directory, all it takes to abort shutdown during execution from a DT-1 terminal is a power down.

When the power is turned back on, the terminal is not doing a shutdown but has a '#' prompt. When the system was inquired as to 'who am i', it replied that it was 'shutdown'. This 'shutdown' has all of the same abilities as 'root'.

The solution to this problem is quite easy. If you wish to leave shutdown as a valid login command, you can modify the /etc/passwd file to execute the /etc/shutdown program only. This, however, gives the end user a choice of how long until the shutdown.

A slightly more complicated solution is to build a file that has appended the number of minutes till shutdown to the /etc/shutdown command. This command would look like this:

```
/etc/shutdown 15
```

Care should be taken to write-protect this file to prevent other users from changing it. If changed, this file could be used by a user to simulate 'root' abilities.

XENIX SHELLS

Below are sections of TRS-XENIX shell programs used by our office to facilitate operations. All of these programs are called as logins and allow other users to perform operations even when a root user is not present. Any of these programs could be assigned a password by the root user with the passwd command. Care must be taken in any of these programs not to allow other users access to change them.

TO LOGIN AS 'save' MAKE THESE CHANGES—

This is a portion of the TRS-XENIX program /usr/bin/save:

```
: Save and restore user filea
: Copyright 1983 by Tandy Corporation
: Tim Dorgsn - 12/82
: This progrsm has been modified to allow any user to
:    request a save from the  console as a login
:    command. It also forces floppy disk verification
:    on.
: - Kraig Snodgrsss - 11/17/83

/etc/verify -f y
PATH=/bin:/usr/bin
if expr "'who am i'" : ".*console.*" > /dev/null
then    : oksy
else    echo "Ssve/Restore must be run from console
        with disk drives"
        exit
fi
```

The /etc/passwd file appears like this to login as save:

```
save::0:0:save procedure:/usr/ssve:/usr/bin/save
```

TO LOGIN AS 'shutdown' MAKE THESE CHANGES—

This is a portion of the TRS-XENIX program /etc/shutdown :

```
:       "arw 3/22/81    shutdown command

                shutdown [time] [su]

     bring the system down gracefully and ruthlessly.
     1. Warn users.
     2. Kill off any user or daemon tssks.
     3. Dismount disks.
     4. sync and halt processor, or go single user.
"
: This program has been modified to allow any user to
:    shut the system down from the console as a login
:    command.
: Krsig Snodgraas - 11/22/83

/etc/verify -f y
PATH=/bin:/usr/bin
if expr "'who am i'" : ".*console.*" > /dev/null
then    : oksy
else    echo "Shutdown must be run from the console"
        exit
fi
```

This is the program /etc/shutdown.15 :

```
: This program is called by a login 'shutdown' and
:    calls and psases a 15 time to /etc/shutdown
: Kraig Snodgrsas - 11/22/83

/etc/shutdown 15
```

The /etc/passwd file appears like this to login as shutdown:

```
shutdown::0:0:shutdown utility
     :/uar/ahutdown:/etc/shutdown.15
```

The following are shell programs I created to allow our office personnel to modify the spooler-printer environment without being root.

TO LOGIN AS 'spoolon' CREATE THIS FILE AS /etc/spoolon

```
: Spoolon login control
: This program will remove the "daemon-lock" from
:    "/usr/spool/lpd" if there is a printer failure
:    of a lock statement. This allows you to restart
:    a printout from the beginning. This program has
:    a nasty habit of trying to hurt itself by trying
:    to kill a pid for the grep. Will give a 'pid#:
:    No such process' but does not affect its
:    operation.
: Kraig Snodgrass - 12/Ø8/83

PATH=/bin:/usr/bin
if expr "' who am i '" : ".*console.*" > /dev/null
then    : okay
else    echo "Spoolon must be run from the console"
        exit
fi

rm /usr/spool/lpd/lock

pids='ps ax | tail +4 | grep ".*lpd.*" | sed -e
     's/ *\([Ø-9]*\) .*/\1/''
if test ! -z "$pids"
then
        echo kill -9 $pids
        kill -9 $pids
        sleep 5
fi

/usr/lib/lpd

echo "Spoolon now completed"
```

The /etc/passwd file appears like this to login as save:

```
spoolon::Ø:Ø:start spool out
     :/usr/spoolon:/etc/spoolon
```

TO LOGIN AS 'spooloff' CREATE THIS FILE AS—/etc/spooloff

```
: Spooloff login control
: This program will leave the spooler on, but prevent
:    output to the printer and will set a lock
:    statement "dummy daemon", if the printer is not
:    on at the time this program is run.
: Kraig Snodgrass - 12/Ø8/83

PATH=/bin:/usr/bin
if expr "' who am i '" : ".*console.*" > /dev/null
then    : okay
else    echo "Spooloff must be run from the console."
        exit
fi

if test -f "/usr/spool/lpd/lock"
then    echo "Spooloff must wait until printer is
        done"
        exit
else    : okay
fi

who am i > /usr/spool/lpd/lock

echo "Spooloff now completed"
```

The /etc/passwd file appears like this to login as spooloff:

```
spooloff::Ø:Ø:stop spool out
     :/usr/spooloff:/etc/spooloff
```

MAGAZINES

Below are nine magazines of special interest to TRS-80 owners that we believe have editorial content of high quality and will be of use to our customers.

Basic Computing—The TRS-80 User Journal
3838 South Warner Street
Tacoma, WA 98409
(206)475-2219

Color Computer Magazine
Highland Hill
Camden, ME 04843
(207)236-9621

Computer User
16704 Marquardt Ave.
Cerritos, CA 90701

80 Micro
P.O. Box 981
Farmingdale, NY 11737

Hot CoCo
P.O. Box 975
Farmingdale, NY 11737

Portable 100—The Magazine for Model 100 Users
P.O. Box 468
Hasbrouck Heights, NJ 07604

PCM—The Portable Computing Magazine
9529 U.S. Highway 42
P.O. Box 209
Prospect, KY 40059

Rainbow (Covers the TRS-80 Color Computer)
P.O. Box 209
Prospect KY 40059
(502)228-4492

two/sixteen magazine
P.O. Box 1216
Lancaster, PA 17603
(717)397-3364

Descending Sort

Bruce Lewis
Box 395
Osceole, MO 64776

I call the following sort a descending sort.

In the first pass, it goes downward from the second item. If the current item is less than the first item, then it is switched with the first item. In the second pass, it goes downward from the third item, and performs switches with the second item, etc.

Here is the subroutine:

```
1ØØØ FOR I=Ø TO N
1Ø1Ø FOR J=I TO N
1Ø2Ø IF ITEM(J) < ITEM (I) THEN EXTRA = ITEM(I)
     : ITEM(I) = ITEM(J)
     : ITEM(J) = EXTRA
1Ø3Ø NEXT J, I
1Ø4Ø RETURN
```

Note: N is the size of the array to be sorted.

PTC-64 Printer Controller

By Bruce Elliott

The PTC-64 Printer Controller (26-1269 $249.95) is a small box which is hooked between your computer and your printer. The computer thinks that the PTC-64 is a printer, and your printer thinks that the PTC-64 is your computer. What the PTC-64 really is, is a second computer. The function of the PTC-64 is to accept printer information as fast as your computer can send it, store the information in PTC-64 RAM, and then feed the information to your printer as fast as the printer can accept it.

WHY WOULD I WANT TO DO THAT?

Several years ago, back in the days when Model IIs were new and printers were generally slower than they are now, I wrote a BASIC program to write a report. The first version of the program would write a line of information to the printer, return, calculate the next line, write it, etc. We had to scrape that version because it took over twelve hours to print the reports. Part of the problem, I'm sure, was my program but most of the time was spent waiting for the printer to print its one line of information so the computer could continue computing. If I had had access to a PTC-64Printer Controller, my programming job would have been over with the first version of the program. The computer could have sent its information to the PTC-64 at high speed, and the PTC-64 would then have fed the printer at its much slower rate.

In general, when your computer sends a character to your printer, the computer must wait for the printer to acknowledge the receipt of that character before the computer can do anything else. Since most printers are slower than the computers they are attached to, this means that your computer has to stop and wait every time it sends information to be printed. Waiting takes time, and means that whatever else the computer could be doing isn't being done.

The PTC-64, with its own Z-80 microprocessor and 64K of RAM, can accept information much faster than most printers. Your computer can send its information to the PTC-64 at a very high speed. The result, from your computer's point of view is that the information has been printed. As the information is being accepted from the computer at high speed, the PTC-64 begins sending the information to your printer as rapidly as your printer can accept it. The net result is that your computer is freed to do more work, and your reports still get printed.

WHAT ELSE CAN THE PTC-64 DO?

Multiple Copies of Documents

Another feature of the PTC-64 is its ability to print multiple copies of any document which is small enough to fit in the PTC-64 memory. If you need three copies of a SCRIPSIT document, you can tell SCRIPSIT to print three copies (and tie up your computer for all the time that printing those copies will take) or you can dump the document into the PTC-64 and have the PTC-64 print the three copies.

While your DWP-210 is working along at top speed making duplicates of the document under control of the PTC-64, you can be using the computer for any other job that you may have.

Special Characters

Another feature of the PTC-64 is its ability to store special characters or character sequences. When the PTC-64 is powered up, it contains definitions for 16 dot matrix characters. If you have a Radio Shack dot matrix printer with bit image capabilities (LP VII, LP VIII, DMP 100, 110, 120, 200, 400, 420, 500, 2100, 2150 or the CGP-220) you can use these special characters, which may not be normally available in your printer.

You can also redefine the characters to suit your needs. Each of the special characters can hold up to 24 bytes of data. The data in the character can be bit image graphic data, ASCII text strings, or any other data that has meaning to your printer.

One example of what can be done is to define a special "character" which prints your name. After the character is defined, a single byte of data to the PTC-64 will send your name (up to 24 characters) to the printer.

Special Drivers

The fact that the PTC-64 is a 64K Z-80 microcomputer suggests that it has powers beyond print buffering.

Imagine your DMP-200 printing gothic characters! Or, maybe you have a version of APL for your computer, and you would like to output your APL listings on the printer using true APL characters.

These are only two of many possible ideas that a special PTC-64 driver could be written to perform.

The manual which comes with the PTC-64 contains the technical information that you will need to program the de-

vice. All you need now is imagination and a knowledge of Z-80 assembly language programming.

While Radio Shack does not currently have any special drivers for the PTC-64, several have been discussed. We will simply have to wait and see what pops out of the product development people.

PTC-64 DETAILS

Status Lamp

The Status Lamp on the PTC-64 gives you a visual indication of the internal status of the printer controller.

Lamp	Status
Off	Buffer empty.
Green	Buffer active, receiving or printing data, or both.
Yellow	Buffer active, less than 4K remaining.
Red	Buffer active and full.

Fault Lamp

Whenever there is a printer problem (for instance, when the printer is Off-Line and you are attempting to print) this lamp will turn on and a beeper (in the PTC-64) will sound eight times. The printer controller will not print or accept data until the fault condition is solved. After the condition is solved, the controller will continue printing where it left off.

CLEAR Key

This key serves a dual purpose:

When the controller is inactive (Status Lamp is OFF), it marks the PTC-64's buffer memory to indicate the starting point for printing and clears any previous buffer contents.

When the controller is active (Status Lamp is ON in any of the three colors), the CLEAR key simply aborts any printing in progress. The information in the buffer remains intact. To start printing again, you must press the COPY key. Printing will then start from the beginning of the buffer. To erase the information in the buffer, press CLEAR a second time.

The CLEAR key function can be duplicated by sending the PTC-64 the control sequence 27 33 in decimal. From BASIC this could be done with:

```
LPRINT CHR$(27); CHR$(33);
```

COPY Key

If the CLEAR key was used to mark the PTC-64 buffer starting point, the PTC-64 will print one copy of the stored information when you press the COPY key. If the text exceeds the buffer size, the COPY key is ignored.

For printing multiple copies, you may also "stack" up to 100 COPY requests by pressing the COPY key repeatedly. Printing of the multiple copies will stop if you press the CLEAR key.

For the PTC-64 to make multiple copies of information, all of the information must fit the PTC-64's buffer.

The COPY key function can be duplicated by sending the PTC-64 the control sequence 27 34 in decimal. From BASIC this could be done with:

```
LPRINT CHR$(27); CHR$(34);
```

PAUSE Key

Pressing this key will temporarily interrupt any printing. Pressing PAUSE a second time will cause printing to resume. If you press PAUSE before sending any information to the PTC-64, the printer controller will accept information from the computer but printing will not start until you press PAUSE the second time.

The PAUSE key function can be duplicated by sending the PTC-64 the control sequence 27 37 in decimal. From BASIC this could be done with:

```
LPRINT CHR$(27); CHR$(37);
```

Multiple Keys

There are three additional functions which can be activated by using the three keys on the PTC-64:

Controls Off—When you press the COPY and PAUSE keys together, the PTC-64 control sequences —special operations to control printing functions - are automatically disabled.

The Controls Off function can be duplicated by sending the PTC-64 the control sequence 27 36 in decimal. From BASIC this could be done with:

```
LPRINT CHR$(27); CHR$(36);
```

Controls On—When you press the CLEAR and COPY keys together, the PTC-64 control sequences are enabled.

Self-Test—When you press the CLEAR and PAUSE keys together, the printer controller automatically performs a self-test, and, if everything is working properly, prints a message on the attached printer.

Additional Functions

Beeper—The PTC-64 half second beeper can be activated using LPRINT CHR$(27); CHR$(42);.

Download—LPRINT CHR$(27); LPRINT CHR$(35); tells the PTC-64 to wait for a new program to be sent from the computer. This allows you to download special drivers into the PTC-64.

Redefine—The 16 special characters that are defined by the PTC-64 can be redefined. To tell the PTC-64 that you are going to redefine these characters, use LPRINT CHR$(27); CHR$(39);.

Translate—The 16 special characters in the PTC-64 are only available after you tell the printer controller to translate the appropriate values as it receives them. If the translate function is off, the 16 codes for the characters will be passed to your printer without translation. To turn translation on use LPRINT CHR$(27); CHR$(38);

Musical Notes

by Bryan Eggers
Software Affair, Ltd.

I KEEP HEARING VOICES!

The Orchestra-90 stereo music synthesizer (#26-1922) allows the transcription and play of any sheet music in up to 5-part harmony, even if you have little or no musical knowledge. So, what is "5-part harmony"? It's five instruments playing at the same time. In music synthesizer language we call it "5-VOICE" harmony. Not singing voices, but instrument voices.

Orchestra-90's VOICES are programmed to play the notes, and the REGISTERS supply the different instrument sounds.

If this is a bit confusing, let's look at it another way. We have 5 musicians (voices), and each is given one instrument (register) to play. How's that?

ORCH-90 has 5 instrument registers: TRUMPET, OBOE, CLARINET, ORGAN, and VIOLIN, referred to by the letters A, B, C, D, and E, respectively. These symbols represent the five possible instrument registers that can be assigned to any voice.

ASSIGNING REGISTERS TO VOICES

Each voice plays notes with register D (organ) unless a different register is assigned to that voice. This is the default voice/register assignment. ORCH-90's "Y" symbol may be used to assign a register to a voice. For example, V4YC assigns register C (clarinet) to Voice 4. After this assignment, all notes played by Voice 4 would sound like a clarinet.

ORCH-90's default voice register assignments look like this:

```
V1YD V2YD V3YD V4YD V5YD
```

If no register assignments occur in the music file, the voices play their programmed notes with the organ (register D) sound.

Here comes the great flexibility of the system. You can switch register assignments anywhere in the music file! Insert a Part Number prior to the measure of music in which the changes are to occur. On the same line, indicate the new voice/register assignments. You can assign different instruments to any or all of the voices at a Part Number. A typical Part Number and register assignment would be:

```
P10 V1YA V2YB V3YE V5YC
```

Voice 1 will now play notes using the TRUMPET sound (register A), Voice 2 uses the OBOE sound (register B), Voice 3 uses the VIOLIN sound (register E), and Voice 5 uses the CLARINET sound (register C).

Voice 4 wasn't changed, so it continues playing with the ORGAN sound (register D) or whatever instrument register was previously assigned to it.

The Part NUMBER itself has no effect on the registers. Using a Part Number alerts the compiler (SCORE command) that voice/register changes may occur.

REPEATS AND REGISTERS

A Part Number also allows us to REPEAT a section of music by simply inserting an "R" followed by the Part NUMBER to be repeated. Let's say that Part 10 contained a dozen measures. Inserting an "R10" at some later point in the music file will cause all measures in "P10" to be repeated. That "P10" could literally contain HUNDREDS of measures and "R10" would repeat them all! The Repeat symbol makes it unnecessary to enter those measures a second time, thus reducing the amount of memory required for the file.

The Repeat symbol instructs ORCH-90's compiler to fetch only the NOTES from the original Part, ignoring all register assignments. The compiler uses the register assignments CURRENTLY in effect at the point in the music file where the Repeat is inserted. This allows you to follow a Repeat symbol with new voice/register assignments:

```
R10 V1YA V2YA V3YC V4YC V5YC
```

You can repeat a Part several times, each with new register assignments:

```
R10 V1YA V2YA V3YA V4YA V5YA
R10 V1YB V2YB V3YC V4YC V5YC
R10 V1YA V2YB V3YD V4YE V5YD
R10 V1YD V2YD V3YA V4YA V5YA
```

This capability will be used for some other tricks when we start defining our own special instruments.

REGISTER DEFINITIONS

What makes the "A" register sound like a trumpet? The answer requires some technical information about the way ORCH-90 creates an instrument waveform. ORCH-90's default registers are based on the spectral analysis ot orchestral instruments and were selected to provide a fairly wide range of sounds. If these registers aren't suitable for a particular piece of music, the registers can easily be altered to create new sounds.

ORCH-90 "builds" a waveform for each instrument. It starts with a fundamental frequency and adds seven partial harmonics, or "partials", to it. Each partial is an integer multiple of the fundamental frequency. The strength or "weight" of each partial determines the overall sound or "tone color" of the register.

ORCH-90's default (internal) definition for the A register (trumpet) is:

```
JASEFA50000E
```

Let's examine this register definition symbol-by-symbol:

SYMBOL	MEANING
J	begin register definition
A	define register A
S	Sinesoidal (harmonic) wave
E	partial #1, fundamental, weight E (nearly maximum)
F	partial #2, first harmonic, weight F (maximum)
A	partial #3, second harmonic, weight A
5	partial #4, third harmonic, weight 5
0	partial #5, fourth harmonic, weight 0
0	partial #6, fifth harmonic, weight 0
0	partial #7, sixth harmonic, weight 0
0	partial #8, seventh harmonic, weight 0
E	overall volume E (nearly maximum)

"J" tells the compiler that we are beginning a new register definition.

"A" is the name of the register being defined (could be A, B, C, D, or E).

"S" indicates Sinesoidal waveform type. ORCH-90 uses this waveform for music. Percussion effects are also possible and may be obtained from either "S" or "R" (Random) waveforms, however, the Percussion Clef interprets the definition in a completely different manner (to be explained in a future column).

The next eight digits represent the weight (strength) of the eight partial harmonics in the waveform. Each of these "partials" may be defined with a weight of 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, or F where 0 is silent and F is the loudest. A setting of 0 indicates that the partial makes no contribution to the final waveform.

The final digit, "E", defines the overall volume of the instrument, relative to the other instrument registers, using the same possible values (0-F) as the partials. A setting of 0 indicates that the entire register will be silent.

ORCH-90 allows each of the A, B, C, D, and E registers to be defined only ONCE within a music file.

Inserting a command like "JASEFA50000E" into the ORCH-90 music file causes the compiler to compare this definition to it's default definition for register A. This comparison takes place each time the music file is SCORED (compiled). In this case, the new definition is identical to the default, so ORCH proceeds normally with the compile. If it were ditterent, the compiler would create a whole new set of frequency tables in memory, adjusted to the new harmonics in the special definition. This is a major task. Fortunately, it only takes a couple of seconds in machine language!

For example, if the definition "JASEFA50000F" were edited into the music file, this would cause the compiler to create new tables. This definition contains a different overall VOLUME of "F" instead of the normal volume of "E" in the default register A. This increases the volume of register A slightly, and even a small change like this requires all new frequency tables. This new definition causes all voices using register A to play notes at maximum volume ("F") instead of the default volume setting of "E".

Each "J" definition added to the tile replaces the setting of the corresponding detault register. Each file may contain only ONE new definition for each register (A, B, C, D, or E). The new definition may be placed ANYWHERE in the file, but the most convenient location is probably at the top of the file. No matter where in the file you insert your new "J" definition, it becomes the effective register definition for the entire file. It more than one "J" definition exists for the same register, the LAST one will be used by the compiler.

DEFAULT REGISTER DEFINITIONS

ORCH-90 creates the following default registers when used on a Model III TRS-80:

JASEFA50000E
JBS48F80000F
JCSE0500000A
JDSF4080000B
JES4F280000D

It you are satisfied with the sound produced by these instrument registers, there is no need to type them into the file. The compiler automatically uses them during a "SCORE". Just use the "Y" command to assign the register name (A, B, C, D, or E) to any voice.

The "J" command is primarily used tor CHANGING the sound or volume of the default instrument registers, allowing each music file to contain a different set of register definitions.

Note that partials 5-8 are not defined in the default registers (they are set to 0). In general, you should not define more than the first four partials if you use ORCH-90 on a Model III TRS-80 (and you should only select the 3 or 4-voice option, not 5-voice). Defining additional partials could cause some undesirable "aliasing". This effect can be reduced somewhat by transposing the piece down with the "<" command. However, if a register is ONLY used to play low bass notes, some additional partials may be added without creating aliasing. It's easy to experiment with the partials. Just go into the EDIT mode, add additional partials (or change the strength of the existing partials), then SCORE and PLAY.

When used on a Model 4 or 4P TRS-80, the ORCH-90 takes advantage of the faster (4 Mhz) clock speed and detines an extra two partials in some of the registers. These are the default register values on a Model 4 or 4P:

```
JASEFA54E00E
JBS48F8F200F
JCSE050F000A
JDSF4080000B
JES4F281400D
```

The faster clock speed of a Model 4 or 4P also allows you to define extra partials. You'll be able to define all the partials (1-8) without aliasing, unless you are trying to play very high notes. Really high pitch notes are obtained by using registers with fewer defined partials. For example, the default D register on a Model 4 can play higher notes than any of the other registers because it has only the first four partials defined. On a Model III, the default C register can play the highest notes because it has only the first three partials defined.

Since partials are actually tones themselves, you can single them out and listen to them individually. For example, try these short files (separately) on a Model 4:

```
EXAMPLE 1
JASF0000000F          define only partial #1 in register A
V1YA                  assign register A to Voice 1
P01                   Part 01
M *Q0,1,2,3,4,5,6,7   plays a scale from Middle-C to ONE octave above Middle-C

EXAMPLE 2
JAS0F000000F          define only partial #2 in register A
V1YA                  assign register A to Voice 1
P01                   Part 01
M *O0,1,2,3,4,5,6,7   plays a scale from ONE octave above Middle-C to TWO octaves above Middle-C

EXAMPLE 3
JAS000F0000F          define only partial #4 in register A
V1YA                  assign register A to Voice 1
P01                   Part 01
M *O0,1,2,3,4,5,6,7   plays a scale from TWO octave above Middle-C to THREE octaves above Middle-C
```

In Example 1 we use only the fundamental tone (partial #1), so ORCH-90 plays the correct notes from Middle-C to one octave above Middle-C.

In Example 2 we eliminate partial #1 and use only partial #2. Partial #2 is twice the frequency of partial #1, therefore the same scale sounds an octave higher than normal.

In Example 3 we eliminate partial #2 and use only partial #4. Partial #4 is four times the frequency of partial #1, therefore it sounds TWO octaves higher than the original scale.

Note that we skipped the 3rd partial. This partial is not an octave multiple of the fundamental tone, so the resulting scale would not be exactly one or two octaves higher. Try it yourself if you're curious.

This demonstrates the individual sounds of various partials, but music played using only single-partial tones is not very interesting (or realistic). No real musical instruments (except ORCH-90 and some other synthesizers) are capable of producing such single-partial tones. All other instruments create multiple-harmonic sounds. In fact, that's what makes them unique. Even an instrument as simple as a kazoo produces several harmonics. ORCH-90's best instrumental effects are obtained using registers with combinations of partials.

BALANCING THE VOICES

Let's get some practical use from our "J" command. You can use the volume parameter (the last digit) to balance the voices, just like in a recording studio. For example, try entering a set of register definitions like this in your music file:

```
JASEFA54E00F
JBS48F8F200B
JCSE050F000B
JDSF4080000B
JES4F281400E
```

Add these voice/register assignments:

```
V1YA V2YB V3YC V4YD V5YE
```

When you transcribe your sheet music, use Voice 1 for the melody because it uses register A (set to highest volume). The melody should be the most prominent voice in the arrangement.

Voices 2, 3, and 4 (all playing with register B) are somewhat lower in volume than the A register, making them perfect for accompaniment. Use these voices for all the harmony and rhythm parts.

Voice 5 (using register E) is slightly less than maximum volume and is perfect for a strong bass line.

No need to limit yourself to the default partial settings when you balance the voices. You could define all the registers as trumpets, yet maintain the same loudness balance:

```
JASEFA54E00F
JBSEFA54E00B
JCSEFA54E00B
JDSEFA54E00B
JESEFA54E00E
```

FADE-OUT ENDINGS

How about a little fade-out ending for your arrangement? Define all the registers with the same partials, but with descending volumes like this:

```
JASEFA54E00F
JBSEFA54E00C
JCSEFA54E007
JDSEFA54E004
JESEFA54E001
```

You now have five trumpets, and each plays at a different loudness level. Next, let's assume that you have a group of measures that you need to repeat at the end of a song. The measures are in Part 80. At Part 80, assign the loudest register (A) to all the voices like this:

```
P80 V1YA V2YA V3YA V4YA V5YA

(measures of music to be repeated go here)
```

ORCH-90 will play all measures in Part 80 (P80) as loud as possible, since all voices are using the loudest register (A). Then, at the first Repeat, assign register B to all the voices. The example now looks like this:

```
P80 V1YA V2YA V3YA V4YA V5YA
(measures of music to be repeated go here)
R80 V1YB V2YB V3YB V4YB V5YB
```

This Repeat (R80) will play all the measures in Part 80 again, except at slightly lower volume. Do the same for the next three Repeats, substituting a softer register each time:

```
P80 V1YA V2YA V3YA V4YA V5YA
(measures of music to be repeated go here)
R80 V1YB V2YB V3YB V4YB V5YB
R80 V1YC V2YC V3YC V4YC V5YC
R80 V1YD V2YD V3YD V4YD V5YD
R80 V1YE V2YE V3YE V4YE V5YE
```

Each time the measures are repeated, the volume will be reduced, yet the SOUND of the instruments never change because you are switching registers with identical partials. You can create even more subtle effects by switching only one or two registers each time instead of all of them. You can also combine these changes with TEMPO changes:

```
R80 V1YB V2YB V3YB V4YB V5YB NO=A0
R80 V1YC V2YC V3YC V4YC V5YC NO=BD
R80 V1YD V2YD V3YD V4YD V5YD NO=CE
R80 V1YE V2YE V3YE V4YE V5YE NO=DF
```

This allows you to simultaneously change the tempo and reduce the volume each time the section of music is repeated.

MUSIC MINUS ONE

If you sing or play an instrument, you'll really like this next trick. ORCH-90 can be used to repeat a song or accompaniment indefinitely (see the "M FILENAME" command), so if you want to practice singing, you'll need to eliminate the melody from the ORCH arrangement and sing that part yourself. Or, perhaps you need to practice singing one of the harmony lines or maybe you want to play the bass guitar while ORCH-90 plays the rest of the file. If you set up the file properly, it is simply a matter of changing ONE character in the file to completely eliminate a particular voice. Define all the registers at the top of the file as we previously did, then assign one to each voice like this:

```
JASEFA54E00F
JBS48F8F200B
JCSE050F000B
JDSF4080000B
JES4F281400E
V1YA V2YB V3YC V4YD V5YE
```

When transcribing your music file, always use Voice 1 for the melody, voices 2, 3 and 4 for harmony parts and Voice 5 for the bass. This sets up your file so that each voice always plays the same musical line throughout the file.

As we mentioned before, the last digit in the "J" definition is the VOLUME of the register. Want to make Voice 1 silent? Just go into the EDIT mode and change the last digit in the A register (the register assigned to voice 1) from F to 0 (no sound)! All notes that were played by Voice 1 in the music file will be silent. You can make any voice silent this way, allowing you to practice any of the five different parts.

ORCH-90 AND THE HAMMOND ORGAN

The partials in a register correspond almost identically to the drawbars of a Hammond organ. This gives you a great source of instrument definitions from countless music books published with Hammond organ settings. You can transfer the values for drawbars 3-9 directly into the ORCH register definition. Use the following chart:

Drawbar (color)	Pitch	Musical Note (from Middle-C)	Scale Name	ORCH Partial Number
3 (white)	8′	Fundamental, or as printed on the music. (C1)	Unison	1
4 (white)	4′	One octave above Unison (C2)	Octave (Eighth)	2
5 (black)	2 2/3′	One octave and five notes above Unison (G2)	Twelfth	3
6 (white)	2′	Two octaves above Unison (C3)	Super Octave (Fifteenth)	4
7 (black)	1 3/5′	Two octaves and three notes above Unison (E3)	Seventeenth	5
8	1 1/3′	Two octaves and five notes above Unison (G3)	Nineteenth	6
9	1′	Three octaves above Unison (C4)	Super Super Octave	8

NOTE: ORCH-90's 7th partial is not available on the Hammond organ, and normally would be set to 0 for these conversions.

For example, suppose that you find a Hammond organ drawbar setting for a Tuba like this:

```
00 8848 136
```

Discard the first two digits (00). They represent Sub-octave and Fifth drawbars (below Middle-C) that are used for very few instruments.

Then, take the next six digits ("884813") and edit them into a register definition like this:

```
JAS884813
```

Add a "0" for ORCH-90's seventh partial:

```
JAS8848130
```

Then add the last digit in the Hammond setting ("6"):

```
JAS88481306
```

Finally, add the overall volume parameter of your choice, for example, "C":

```
JAS88481303C
```

There you have it. TUBA for ORCH-90. Of course, if you really want ORCH-90 to sound like a particular instrument, you must take other factors into consideration. For example, think about the range of notes playable by the instrument. Tuba players don't play "Flight of the Bumblebee". There are certain physical limitations to consider, including note durations and articulation. You can consult music theory books for the typical range of various instruments. Look for sheet music scored for specific instruments. Chances are the arranger already took these characteristics into account when the score was written.

NEW ORCH-90 INSTRUMENT REGISTERS

Here are register definitions for some traditional instruments:

JASA2840000F / CHIMNEY FLUTE
JAS00020004F / CONCERT PICCOLO
JASE2020000F / FOREST FLUTE
JASE8440000F / GERMAN FLUTE
JASE2420000F / MAGIC FLUTE
JASCEEA0402F / MELOPHONE
JAS66600000F / MELODIA
JAS8E480402F / SOLO OPEN FLUTE
JAS46802200F / VIENNA FLUTE
JAS48EA6402F / GRAND VIOLIN
JAS08C64602F / ORCHESTRAL VIOLIN
JAS68C60202F / 'CELLO
JAS48A20206F / VIOLA D' AMORE CELESTE
JASACE20402F / VIOLIN
JAS82C44206F / VIOLINCELLO
JAS0022420AF / APFELREGAL
JASA6884402F / BALLAD HORN
JAS44A62406F / BAROQUE OBOE
JAS8AAAAA08F / BAROQUE TRUMPET
JASC0A22202F / BASSOON
JASA2E26608F / BELL CLARINET
JASACEF8604F / BUGLE
JAS0A0A0A08F / CLAIRION HARMONIQUE
JASA0C00602F / CLARINET
JAS04240200F / CONTRA OBOE
JAS20286402F / DULZIAN
JAS262A260CF / EGYPTIAN BAZU
JAS84E82408F / ENGLISH HORN
JASAFEF8604F / FLUGELHORN
JASEACC0000F / FRENCH HORN
JASCAA20200F / HUNTING HORN
JASEFFCFF0FF / MILITARY FANFARE
JAS20AE0800F / MUSETTE MIRABILIS
JASAAEE4406F / MINOR TRUMPET
JAS02A20806F / ORIENTAL REED
JAS8EAEEE0EF / POST HORN
JAS42048A06F / ROHRSCHALMEI
JASA6220000F / SACKBUT
JASFC600402F / SAXOPHONE
JASAE8C080EF / SILVER TRUMPET
JASFF8FFF0FF / SOLO TUBA
JASCFCE6000F / STENTOR HORN
JASEFFEA602F / TROMBONE
JASCEE22200F / WALDHORN
JAS26ACCC04F / BANJO
JAS40060004F / BELLS
JAS26020202F / CELESTA HARP
JAS2C020002F / CHIMES
JAS02000002F / GLOCKENSPIEL
JAS48C88402F / GUITAR
JASA4022240F / HARP
JAS600C0000F / MARIMBA
JASC4020600F / MARIMBA HARP
JAS24040008F / ORCHESTRAL BELLS
JAS6F220002F / VIBRA-HARP
JAS0A0A0E0EF / ZINK
JAS20022202F / VOX HUMANA
JAS46A24404F / GAMBA
JASC8422200F / CLARABELLA
JAS26ACCC0CF / TOY TRUMPET
JAS26E06402F / SARRUSOPHONE
JASFFFFFF0FF / OPHECLEIDE
JASFCA00000F / SOLO SAXOPHONE
JASC0C0A202F / MELODY CLARINET
JAS0E080400F / CORNO OCTAVE
JAS20040008F / MINIATURE BELLS
JAS28222202F / ETHEREAL VIOLINS
JASA8642202F / PIANO

On the other hand, who cares about traditional instruments? Since ORCH-90 lets you create your own unique sounds, perhaps you'll invent some interesting new registers that we can share with other users. I'm putting together a list of ORCH-90 special effects, both musical and percussion, for a future article. Please send any new instrument definitions, special effects or questions to my attention, in care of TRS-80 Microcomputer News or leave a message on our ORCH-90 SIG (page HOM-13) on CompuServe. We have a few hundred music files available FREE in the CompuServe ORCH-90 SIG database. Check in sometime and learn a few tricks from the local experts or, even better, teach us a few techniques of your own!

CORRECTION

Musical Notes
February, 1984

These corrections are for a mistake in the February "Musical Notes" column. Due to a last minute change in the ORCHUTIL/CMD program (prior to production), the patches to make ORCHUTIL/CMD accept job files were incorrect. The INCORRECT patches were:

PATCH UTIL/CMD (ADD=5A9D,FIND=9554, CHG=0000)
PATCH UTIL/CMD (ADD=5AA8,FIND=06, CHG=0C)
PATCH UTIL/CMD (ADD=5AC1,FIND=2430, CHG=2B00)

The CORRECT patches for ORCHUTIL/CMD, version 01.01.00 are:

PATCH UTIL/CMD (ADD=74A2,FIND=956E, CHG=0000)
PATCH UTIL/CMD (ADD=74AD,FIND=06, CHG=0C)
PATCH UTIL/CMD (ADD=74C6,FIND=2430, CHG=2B00)

Printer Graphics

Paul Brannon
201 Massee Dr.
Dothan, Ala. 36301

This program will print graphic characters using Radio Shack's own Upper/Lower case routine supplied with the Model I Double Density TRSDOS. It might work on the cassette version, using the same idea. The program replaces the lowercase mode (SHIFT 0) with a graphic characters mode.

First,load the program and RUN it. To access the graphic character mode, (SHIFT) (0). A through Z are graphic characters 129 to 154, SHIFT A through SHIFT Z are 155 to 180, and 1 through 9 create 181 to 189. 190 and 191 are found by pushing "," and ".". To exit graphic mode, (SHIFT) (0).

The program is so simple, you can easily make any changes to it to suit your needs. Also, you can use it to change (SHIFT) (@) to other special symbols other than the British Pound sign. Furthermore, you can enter any BASIC word with one press of a button in graphics mode. Example—RUN equals graphic code 142. See Section E in the Level II manual to see which word equals which graphic code.

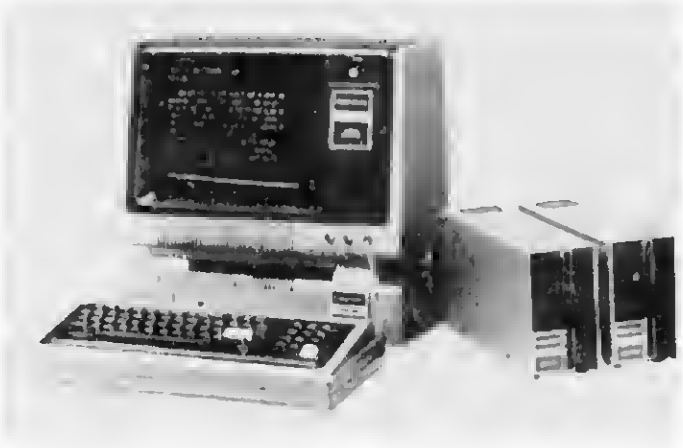
```
1Ø A=129
     : B=155
     : C=181
2Ø FOR X=&HFFØØ TO &HFF19
     : POKE X, A
     : A=A + 1
     : NEXT X
3Ø FOR X=&HFF4Ø TO &HFF59
     : POKE X, B
     : B=B + 1
     : NEXT X
4Ø FOR X=&HFF2Ø TO &HFF28
     : POKE X, C
     : C=C + 1
     : NEXT X
5Ø POKE &HFF2B, 19Ø
     : POKE &HFF2O, 191
```

Fibonacci Sequence Generator

Martin Combs
Verde Valley School
Sedona, AZ 86336

Here is a program for readers who like to crunch large numbers. It was done on a 16K TRS-80 Model I.

```
1Ø REM FIBONACCI SEQUENCE GENERATOR
2Ø REM BY HARTIN COMBS, VEROE VALLEY SCHOOL, SEDONA
     AZ 86336
3Ø REM FIBONACCI SEQUENCES ARE GENERATEO RECURSIVELY
4Ø REM EACH TERM IS THE SUM OF THE TWO PREVIOUS
     TERMS.
5Ø REM FOR "THE" FIBONACCI SEQUENCE THE FIRST TWO
     TERMS
6Ø REM ARE BOTH 1, BUT OTHER SEQUENCES ARE POSSIBLE.
7Ø REM THIS PROGRAM WILL ONLY START WITH SINGLE
     OIGITS
8Ø REM TO SEE RESULTS, HIT BREAK, TYPE IN F=1, THEN
     CONT.
9Ø REM TO USE PRINTER, HIT BREAK, TYPE IN F=2, THEN
     CONT.
1ØØ REM THE PROGRAM RUNS FASTER, BUT INVISIBLY, IF
     F=Ø.
11Ø DEFINT A-Z
     : OIM A(1ØØØ), B(1ØØØ), S(1ØØØ)
     : REM 1ØØØ OIGITS ARE PLENTY, BUT YOU CAN HAVE
     MORE.
12Ø PRINT "WHAT POSITIVE INTEGER LESS THAN 1Ø DO YOU
     WANT"
13Ø INPUT "AS THE FIRST TERM OF THE FIBONACCI
     SEQUENCE"; A(1)
14Ø PRINT "WHAT POSITIVE INTEGER LESS THAN 1Ø DO YOU
     WANT"
15Ø INPUT "AS THE SECONO TERM"; B(1)
16Ø K=3
     : REM K IS THE NUMBER OF THE TERM IN THE
     SEQUENCE
17Ø D=2 + INT(K / 4.78)
     : REM D IS ALWAYS 1 OR 2 MORE THAN THE NUMBER OF
     DIGITS IN THE TERM BEING CALCULATEO.
18Ø FOR I=1 TO O
19Ø S(I)=A(I) + B(1) + C
     : REM THE RECURSIVE FORMULA. C IS THE CARRY.
2ØØ C=Ø
     : REM REINITIALIZE THE CARRY.
21Ø IF S(I)>9 THEN S(I)=S(I) - 1Ø
     : C=1
22Ø NEXT
23Ø IF F=Ø GOTO 31Ø
     : REM SUPPRESSES THE OUTPUT TO INCREASE SPEEO.
24Ø FOR I=O TO 1 STEP -1
     : REM LOOP PRINTS RESULTS.
25Ø IF S(I)=Ø AND G=Ø GOTO 28Ø
     : REM LINES 25Ø, 26Ø, 3ØØ SUPPRESS LEADING ZEROS
     IN OUTPUT.
26Ø G=1
27Ø IF F=1 THEN PRINT RIGHT$(STR$(S(I)),1); ELSE
     LPRINT RIGHT$(STR$(S(I)), 1);
28Ø NEXT
29Ø IF F=1 THEN PRINT ELSE LPRINT
3ØØ G=Ø
31Ø K=K+1
     : FOR I=1 TO D
     : A(I) = B(I)
     : B(I) = S(I)
     : NEXT
     : REM NEWLY CALCULATEO TERMS BECOME INPUTS FOR
     NEXT TERM.
32Ø IF F=Ø THEN PRINT "FIBONACCI TERM #"; K; "IS
     BEING CALCULATEO" ELSE IF F=1THEN PRINT
     "FIBONACCI TERM #"; K; "IS:" ELSE LPRINT
     "FIBONACCI TERM #"; K; "IS:"
33Ø GOTO 17Ø
```



Elementary Math Generator

Paul J. Breaux
1033 NW 22nd Street
Moore, OK 73160

When I was thinking of purchasing a computer, I had to convince the wife that it could be used to help the family. One of the items I cited was in the education of our children. This was one of my first programs in BASIC, which is a computer language I had never used in programming.

The math program is very simple to use and modify. It was originally implemented for multiplication problems, however, by changing the sign in line 240, it could be used for addition as well as subtraction problems.

The programs will display the problem to be solved as well as the amount of right and wrong answers accumulated. This will assist both the student and the teacher in knowing how the student is progressing.

The computer will first request the student's name. Then there will be a request for the multiplicand and next the multiplier. If, for example, the student is practicing the three multiplication table, you could put ten when the first number is requested and three when the second number is requested. This will insure that the numbers zero to ten are multiplied by zero to three.

The program will also insure that the higher number will always be the multiplicand or top number in the problem. When the student answers the problem, a flashing message at the top of the display will advise the student if the problem was answered correctly or incorrectly. The same problem will be displayed until the student answers correctly.

I am sure that there may be other ways to more personalize the program to suit everyone's requirements. Various types of rewards could be displayed on the monitor such as proceed to next class after completion of a certain number of problems. The uses and possibilities for program modifications are left to your imagination. Happy programming.

```
5 REM MULTIPLICATION PROGRAM FOR ELEMENTARY STUDENTS
6 REM PROGRAMMER: PAUL J. BREAUX
7 REM DATED: 1 JULY 1980
90 REM THIS IS AN ELEMENTARY MATH GENERATOR AND TEST
100 CLS
101 INPUT "WHAT IS YOUR NAME"; M$
102 CLS
110 INPUT "ENTER FIRST NUMBER"; X
120 CLS
130 INPUT "ENTER SECOND NUMBER"; Y
149 CLS
150 A=RND(X)
160 B=RND(Y)
161 N=N * 0
170 PRINT @ 104, "WRONG", W
180 PRINT @ 448, "PROBLEM"
190 PRINT @ 168, "RIGNT", R
191 IF B > A THEN GOTO 510
200 PRINT @ 468, A
210 PRINT @ 591, "X"
211 IF A < 10 THEN GOTO 220
212 IF A > 99 THEN GOTO 216
213 IF B > 9 THEN GOTO 220
214 PRINT @ 597, B
215 GOTO 230
216 IF B > 10 THEN GOTO 219
217 PRINT @ 598, B
218 GOTO 230
219 IF B < 100 THEN GOTO 214
220 PRINT @ 596, B
230 INPUT "ANSWER"; Z
240 G = A * B
250 IF G <> Z THEN GOTO 400
260 R = R + 1
270 GLS
280 IF N = 3 THEN GOTO 150
290 PRINT "THAT IS THE RIGNT ANSWER "; M$
300 FOR F = 1 TO 100
310 NEXT F
320 GLS
330 FOR F = 1 TO 100
340 NEXT F
350 N = N + 1
360 GOTO 270
400 W = W + 1
410 GLS
420 IF N = 3 THEN GOTO 161
430 PRINT "THAT IS THE WRONG ANSWER "; M$
440 FOR F = 1 TO 100
450 NEXT F
460 GLS
470 FOR F = 1 TO 100
480 NEXT F
490 N = N + 1
500 GOTO 410
510 F = 0
511 G = 0
512 F = A
513 G = B
514 A = 0
515 B = 0
516 A = G
517 B = F
518 GOTO 200
520 PRINT @ 591, "X"
530 PRINT @ 596, A
540 GOTO 230
```

Math Text Writer

Paul J. Breaux

My daughter brought home some math problems that the teacher had written out manually. I could imagine how long this took with twenty students in her class. Being a programmer by profession, I wrote a program that will write math problems using a printer and random number generator. I figure that this would assist teachers as well as parents who may think their children need more practice in solving simple math problems.

This computer program is very simple to use and modify. The program will print out single digit problems for addition, subtraction, and multiplication depending on what mathematical sign is entered when requested by the program. The program will also request that the user enter the number of problems in multiples of eight.

This program was designed using the TRS-80 Model I Level II with a Line Printer VII. Modifications may have to be made, depending on the printer you use. The program can also be modified to print multi-digit numbers. By changing the random number function to a higher number and modifying the PRINT statements, you will be able to print multi-digit numbers.

This program will assist teachers in writing more problems outside the assigned textbook for students that require further practice. It can also be modified to print the answers to the problems at the end of the program. This will assist in checking answers faster and relieve the teacher/parent to do other tasks as required.

My daughter has increased her skills considerably. She also ask me to run off some extra copies so that when she and her friends play school they can use the listings as a math test. After all, that is what using a computer for school is all about. By making it fun, hopefully the students will pick up material faster, and retain more of what they've learned.

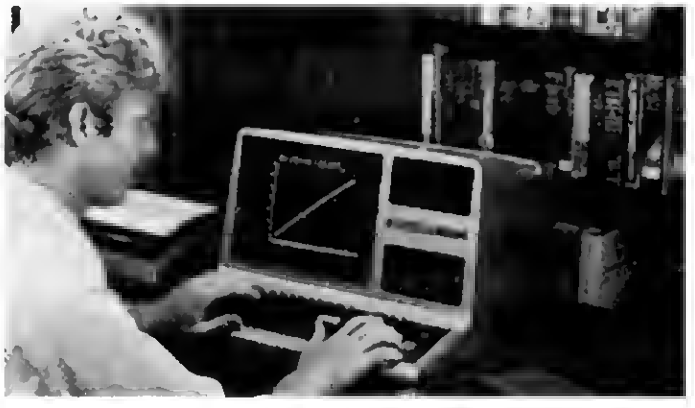
```
5 REM **** MATH TEXT WRITER ***********
6 REM **** DATED: 29 NOV 81 ***********
7 REM **** PROGRAMMER: PAUL J. BREAUX *
8 CLS
     : PRINT "THIS IS THE MATH TEXT WRITER FOR
     ADDITION, SUBTRACTION, "
9 PRINT "AND MULTIPLICATION PROBLEMS. THIS IS FOR
     SINGLE DIGIT NUMBERS ONLY.";
10 INPUT " WHAT SIGN DO YOU WANT TO PUT ON PROBLEMS";
     S$
15 CLS
     : INPUT "NUMBER OF PROBLEMS IN MULTIPLES OF
     EIGHT, TOTAL OF 80 PROBLEMS"; P
20 CLS
     : PRINT "PRINTING PROBLEMS";
30 N=0
     : D=0
35 C=0
     : D=D + 1
40 N=N + 1
50 AA=RND(9)
60 C=C + 1
70 LPRINT N ".  ";
     : LPRINT AA;
80 IF C<>8 THEN GOTO 40
90 LPRINT CHR$(13)
100 X=5
     : Y=0
     : IF D=2 THEN GOTO 110
105 IF N>=10 THEN X=6
110 AA=RND(9)
115 IF S$="-" THEN AA=RND(5)
120 LPRINT TAB(X) S$ AA;
130 X=X + 9
131 IF N>=10 THEN X=X + 1
140 Y=Y + 1
150 IF Y<>8 THEN GOTO 110
160 LPRINT CHR$(13)
161 IF N>=10 THEN GOTO 190
180 LPRINT "      ---      ---      ---      ---
     ---      ---      ---      ---"
181 GOTO 230
190 LPRINT "      ---      ---      ---      ---
     ---      ---      ---      ---"
230 LPRINT
     : LPRINT
235 IF N=P THEN GOTO 5
240 GOTO 35
```

Video Graphic Reverser

Steve Banfield
6042 N. Lee #4G
Morrow, GA 30260

Here is a program that will reverse all the graphic characters on a Model I or III screen. All white dots will be black and vice-versa.

```
10 '*** VIDEO GRAPHIC REVERSER ***
20 '*** STEVE BENFIELD 8/16/81 ***
30 '*** 6042 N. LEE ST 4G      ***
40 '*** MORROW, GA 30260       ***
50 ' REVERSES ALL GRAPHICS
60 FOR I=15360 TO 16383
70 Z9=PEEK(I)
     : IF Z9=32 THEN Z9=128
80 Z9=Z9 - 128
     : IF Z9<0 THEN 100
90 POKE I, 191 - Z9
100 NEXT I ': RETURN if subroutine
```



Screen Print

Nelson C. Haldana
Mathematics Dapartment
Jackson High School
501 Argonaut Lana
Jackson, CA 95642

I use the following program with my Model I to print on the printer what is displayed on the screen. I add lines 64999 to 65011 to the end of my main program and use a GOSUB 65000 in the main program at points where I might want to have hardcopy. The program can also be stored on a disk (without line numbers 64999, 65002, and 65011) and when hardcopy is desired, a RUN "filespec" is used.

```
64999 END
65000 C=0
     : INPUT "DO YOU WANT HARDCOPY (Y/N)"; H$
     : PRINT CHR$(29), CHR$(27), CHR$(27)
65001 IF H$="Y" THEN INPUT "TURN ON PRINTER - HIT
     <ENTER> WHEN READY"; P$
      : PRINT CHR$(29), CHR$(27), CHR$(27)
65002 IF H$="N" THEN RETURN
65003 INPUT "FIRST LINE TO PRINT"; L1
     : PRINT CHR$(29), CHR$(27), CHR$(27)
65004 INPUT "LAST LINE TO PRINT"; LL
     : PRINT CHR$(29), CHR$(27), CHR$(27)
65005 SL=15360 + ((L1 - 1) * 64)
65006 EL=15360 + ((LL - 1) * 64) + 63
65007 FOR X=SL TO EL
65008 LPRINT CHR$(PEEK(X));
65009 C=C + 1
     : IF C=64 THEN LPRINT CHR$(10)
     : C=0
65010 NEXT X
65011 RETURN
```

Eat

Jeff Klug
1040 W. Notre Dame Dr.
Altamonte Spgs. FL 327011

For this Model I/III program, use the arrow keys for movement. The object is to try to run into the dots.

```
1Ø 'EAT -- BY JEFF KLUG Ø7/22/82
2Ø CLEAR 5Ø
     : DEFINT A-Z
     : DEFDBL T
     : CLS
3Ø DOT$=CHR$(14Ø) + CHR$(179) + CHR$(14Ø)
4Ø UP$=CHR$(191) + CHR$(176) + CHR$(191)
5Ø DN$=CHR$(191) + CHR$(131) + CHR$(191)
6Ø RT$=CHR$(191) + STRINC$(2, 179)
7Ø LF$=STRINC$(2, 179) + CHR$(191)
8Ø PT$="POINTS:"
     : TM$="TIME:"
     : TIME 1Ø1
     : AM=475
     : GY$=UP$
9Ø PRINT @ DOT, STRING$(3, 128),
     : COUNT=21
     : DOT=RND(956) + 64
1ØØ A=PEEK(144ØØ)
     : TIME=TIME - .25
     : IF TIME<Ø THEN 2ØØ ELSE COUNT=COUNT - 1
     : IF COUNT<1 THEN GOTO 9Ø
11Ø IF A=8 THEN BM=-64
     : GY$=UP$
     : GOTO 15Ø
12Ø IF A=16 THEN BM=64
     : GY$=DN$
     : GOTO 15Ø
13Ø IF A=32 THEN BM=-3
     : GY$=LF$
     : GOTO 15Ø
14Ø IF A=64 THEN BM=3
     : GY$=RT$
     : GOTO 15Ø
15Ø PRINT @ AM, STRING$(3, 128);
     : IF AM + BM < 64 OR AM + BM > 1Ø2Ø THEN AM=AM
    ELSE AM=AM + BM
16Ø PRINT @ AM, GY$;
     : PRINT @ DOT, DOT$;
17Ø FOR F=-2 TO 2
     : IF AM=DOT + F THEN PT=PT + 5Ø
     : COUNT=1
     : NEXT F ELSE NEXT F
18Ø PRINT @ Ø, PT$; PT;
     : PRINT @ 32, TM$; INT(TIME);
19Ø GOTO 1ØØ
2ØØ CLS
     : PRINT CHR$(23);
     : PRINT @ 476, "THE END";
21Ø PRINT @ 512, PT$; PT;
22Ø PRINT @ 768, "";
     : END
```

Printer Font

Stephen J. Rossello
432 New Boston Street
Canastota, NY 13032

You may be interested in the following program which allows the operator to choose a font for the line printer (I have a Line Printer IV) then automatically go into SCRIPSIT. I normally execute the program, labeled "FONT", from an AUTOed DO file called STARTER/BLD. The commands in the file are:

```
BASIC FDNT -F:Ø
```

This program was written to work with Tape SCRIPSIT moved to disk, and a Line Printer IV. If you want to use it with another program, or a different printer you will need to modify the program.

```
3Ø CMD "B", "OFF"
1ØØ CLS
     : PRINT TAB(28) "SCRIPSIT"
     : PRINT
     : PRINT
     : PRINT "Choose the font you want:"
     : PRINT
     : PRINTTAB(5) "<1> Normal type - 1Ø
     Characters/inch, 8Ø characters/line"
     : PRINT TAB(1Ø) "maximum."
     : PRINT
     : PRINT TAB(5) "<2> Compressed type - 16.5
     characters/inch,"
1Ø1 PRINT TAB(1Ø) "132 characters/line maximum."
     : PRINT
     : PRINT TAB(5) "<3> Proportional type -
     variable, approximately"
     : PRINT TAB(1Ø) "13 characters/inch, 1ØØ
     characters/line."
     : PRINT TAB(1Ø) "(Do not use with
     justification.)"
11Ø I$=INKEY$
     : IF I$="" GOTO 11Ø
111 I=VAL(A$)
     : IF I<1 OR I>3 GOTO 11Ø
12Ø ST%=PEEK(14312) AMD 24Ø
     : IF ST%<>48 THEN PRINT @ 96Ø, "LINE PRINTER NDT
     READY";
121 ON I GOSUB 13Ø, 131, 132
     : LPRINT CHR$(27); CHR$(138)
     : CMD "B", "DN"
     : CMD "L", "SCRIPS/CMD"
122 USRØ=$H99CØ
     : J=USRØ(Ø)
13Ø LPRINT CHR$(27); CHR$(19)
     : RETURN
131 LPRINT CHR$(27); CHR$(2Ø)
     : RETURN
132 LPRINT CHR$(27); CHR$(17)
     : RETURN
```

Video Weaver

D.A. Goldman

This program is designed to create different patterns on the Model I/III screens. If you don't like the pattern, just press any key (like the space bar) to start again. Once you get the hang of it, try changing the values for XA, XB, YA, and YB in line 30 (these are the minimum and maximum values that x and Y can reach.) As a sample, try:

```
3Ø XA=2Ø: XB=1ØØ: YA=1Ø: YB=4Ø
```

PROGRAM

```
1Ø REM ** VIDED WEAVER ** BY D.A. GOLDMAN
2Ø DEFINT A-Z
     : RANDOM
```

```
30 CLS
     : XA=0
     : XB=127
     : YA=0
     : YB=47
40 DX=SGN(RND(2)-1.5)
     : DY=SGN(RND(2)-1.5)
50 X=RND(XB - XA) + XA
     : Y=RND(YB - YA) + YA
60 IF X + DX > XB OR X + DX < XA THEN DX=-DX
70 IF Y + DY > YB DR Y + DY < YA THEN DY=-DY
80 X=X + DX
     : Y=Y + DY
90 IF PDINT(X, Y) THEN RESET(X, Y)
     : RESET(XB - X, Y)
     : RESET(XB - X, YB - Y)
     : RESET(X, YB - Y) ELSE SET(X, Y)
     : SET(XB - X, Y)
     : SET(XB - X, YB - Y)
     : SET(X, YB - Y)
100 IF INKEY$="" THEN 60 ELSE 30
```

Rectangle Draw

Grady Koch
2929 Duke of York Dr.
Chesapeake, VA 23321

This Model III program randomly picks a point and draws a rectangle there of a certain size. A FOR-NEXT loop allows a number of rectangles to be drawn.

```
5 'RECTANGLE DRAW BY GRADY KOCH
10 CLS
20 INPUT "INPUT THE NUMBER OF RECTANGLES
TO BE DRAWN"; A
30 CLS
40 FOR S=1 TO A
50 B=RND(112)
60 C=RND(32)
70 D=RND(15)
80 SET(B, C)
90 FOR E=B TO B + D
     : SET(E, C)
     : NEXT E
100 FDR F=C TO C + D
     : SET(B + D, F)
     : NEXT F
110 FOR G=B TO B + D
     : SET(G, C + D)
     : NEXT G
120 FOR H=C TO C + D
     : SET(B, H)
     : NEXT H
130 NEXT S
140 GOTO 140
```

Quick Label II

David Young
6355 Robinhood Lane
Merriam, KS 66203

Here is a program for a Model III and Line Printer IV. It is similar to the program by Mr. MacLean, except that it uses LINE INPUT statements rather than INPUT statements.

LINE INPUT statements allow the use of commas, quotation marks, semicolons, or other delimiters which might be needed in making labels addressed to companies or labels with commas in the address line after the city.

```
10 CLEAR 100
20 CLS
30 INPUT "HOW MANY COPIES OF THIS LABEL"; C
40 PRINT
50 LINE INPUT "ENTER NAME                  ? "; N$
60 LINE INPUT "ENTER ADDRESS               ? "; A$
70 LINE INPUT "ENTER CITY, STATE AND ZIP   ? "; S$
80 PRINT
90 PRINT
100 PRINT "PRESS ANY KEY IF THE INPUT DATA IS
    CORRECT, ELSE BREAK"
110 IF INKEY$="" THEN 110 ELSE 120
120 FOR Z=1 TO C
130 LPRINT
140 LPRINT TAB(5) N$
150 LPRINT TAB(5) A$
160 LPRINT TAB(5) S$
170 LPRINT
180 LPRINT
190 NEXT Z
200 GOTO 20
```

Repeating Keys

Chandra Bajpai
180 Zabrosky Rd
Stoughton, MA 02072

For those of you who have a Model III and want to take advantage of the Model III's repeating keys, type in the following program. This program was made as a keyboard input for a screen editor I was making, but it can be adapted for other programs. I wrote this program using my 48K two disk Model III.

Line 40 sets up a string containing the machine language program.

Lines 50 - 80 set up for a USR call.

Line 90 prints a fake cursor.

Line 100 calls the machine language program.

Line 110 checks to see if the BREAK key has been pressed.

Line 120 checks to see if the CLEAR key has been pressed.

Line 130 prints the character and goes to line 110 to get another key.

```
10 REM MACHINE LANGUAGE KEYBDARD INPUT UTILITY
20 REM BY C. BAJPAI 1982
30 DATA 3B, 0, 205, 73, 0, 111, 195, 154, 10
40 FOR X=1 TO 9
     : READ B
     : S$=S$ + CHR$(B)
     : NEXT
50 X=VARPTR(S$)
     : S1=PEEK(X + 1)
     : S2=PEEK(X + 2)
60 S0=S1 + S2 * 256
70 IF S0>32767 THEN S0=-1 * (65536 - S0)
80 DEFUSR1=S0
90 CLS
     : PRINT CHR$(143);
100 KEY=USR1(0)
110 IF KEY=1 THEN STOP
120 IF KEY=31 THEN 90
130 PRINT CHR$(8); CHR$(KEY); CHR$(143);
     : GOTO 100
```

ASSEMBLY LANGUAGE LISTING

```
Hex             Z8Ø op code

26ØØ            LD H, Ø
CD49ØØ          CALL ØØ49 ($KBWAIT)
6F              LD L, A
C39AØA          JP ØA9Ø
```

Audio Subroutine

David G. Blood
5122 Arquilla Drive
Richton Park, IL 60471

The following program is for a Level II Model III.

This subroutine turns the cassette recorder on so that verbal instructions can be added to a program, or any other audio sounds may be activated when the subroutine is called. The subroutine also turns the recorder off after the specified number of seconds have elapsed.

This simple subroutine can be used to add mysterious sounds to an adventure program, give verbal instructions to slow readers, or even add a musical break in the middle of long drill and practice programs.

The subroutine is very easy to use. When typing in the program, simply add GOSUB 500 at the point you wish to activate the cassette recorder.

After saving the program on tape, unplug all lead wires that go to the computer and manually record, in sequential order, the audio output you wish the user to hear.

When using a program with this subroutine in it, the black load plug going from the cassette recorder to the computer must be unplugged after the tape is loaded. An earplug may or may not be used, depending on the situation.

The structure of the program is very simple. Line 10 sends the program to the subroutine. Line 500 is a counter, if X=60 then 500 will keep the recorder on for 60 loops or about 30 seconds. Line 501 is an error handler to take care of the lack of input that line 502 will encounter. There will not be any data input because the black load plug has been unplugged. Line 503 is the other half of the error handler statement and starts the loop all over again. Lines 504 and 505 turn the cassette off. And line 506 returns the program to the main calling routine.

```
1Ø GOSUB 5ØØ
5ØØ X=X + 1
     : IF X=6Ø THEN 5Ø4
5Ø1 ON ERROR GOTO 5Ø3
5Ø2 INPUT #-1
5Ø3 RESUME 5ØØ
5Ø4 POKE 16526, 248
     : POKE 16527, 1
5Ø5 P=USR(Ø)
5Ø6 RETURN
```

Attack-Man

Robert Patton
Rt 3, Box 449
High Point, NC 27263

This game program was written on a Model III using Disk BASIC.

```
1Ø CLS
11 M=RND(964)-1
     : D$=CHR$(153)
     : PRINT @ M, D$
2Ø H=Ø
21 J=64
22 K=128
25 H=H + 5
     : CLS
26 IF H>832 GOTO 1Ø
27 IF J>896 GOTO 1Ø
28 IF K>96Ø GOTO 1Ø
29 PRINT @ M, D$
     : M=M + 3
     : IF M>1Ø23 GOTO 11
3Ø PRINT @ H, CHR$(176) + CHR$(188) + CHR$(191) +
   CHR$(191) + CHR$(191) + CHR$(179) + CHR$(191) +
   CHR$(143)
     : IF H=M GOTO 15Ø
45 J=J + 5
     : IF J=M GOTO 15Ø
5Ø PRINT @ J, STRING$(4, 191) + STRING$(2, 188) +
   CHR$(176)
     : IF J=M GOTO 15Ø
65 K=K + 5
     : IF K=M GOTO 15Ø
7Ø PRINT @ K, CHR$(128) + STRING$(7, 131)
     : IF K=M GOTO 15Ø
8Ø GOTO 25
15Ø PRINT "TOO BAD, DO YOU WANT TO GO AGAIN (Y OR N)"
155 Z$=INKEY$
     : IF Z$="" GOTO 155 ELSE IF Z$="Y" GOTO
     1Ø ELSE IF Z$="N" THEN ENO ELSE 15Ø
```

Word Scramble

David Risack
99 Stephenville Pky.
Edison, NJ 08820

When you feel a little scrambled, have fun scrambling your brain with "Word Scramble." This Model III Level II program will take two people, one to type the word and one to guess the word. I have solved the problem of scrambling words, by taking one letter at a time and randomly putting them into an array.

```
1 ON ERROR GOTO 1ØØØ
1Ø CLS
     : INPUT "TYPE IN WORD TO BE SCRAMBLED"; A$
     : CLS
2Ø A=LEN(A$)
     : OIM A(A)
3Ø FOR X=1 TO A
     : B=ASC(MID$(A$, X, 1))
4Ø RANDOM
     : N=RND(A)
     : IF A(N)<>Ø THEN 4Ø ELSE A(N)=B
5Ø NEXT X
6Ø FOR X=1 TO A
7Ø PRINT CHR$(A(X));
8Ø NEXT X
9Ø PRINT
     : PRINT
1ØØ INPUT "GUESS WHAT WORD IT IS "; B$
     : IF A$=B$ THEN PRINT "THAT'S RIGHT"
     : FOR X=1 TO 1ØØØ
     : NEXT X
     : RUN ELSE PRINT "TRY AGAIN"
     : GOTO 6Ø
1ØØØ IF ERR / 2 + 1=14 PRINT "* * * SORRY YOUR WORD
     CANNOT BE LONGER THEN PROVIOED STRING SPACE * *
     *"
     : FOR X=1 TO 12ØØ
     : NEXT X
1Ø1Ø RUN
```

Scrip for the PC-2

Marion F. Brown
P.O. Box One
Marlow, NH, 03456.0001

(Editor's Note: Mr. Brown's program not only demonstrates some of the excellent and sometimes surprising features of the PC-2, but also provides some marvelous examples of what can be accomplished with the PC-2 printer. The program has been faithfully reproduced as it was submitted. We have tried to present the information regarding the program as clearly as possible although we were unable to follow fully some of the more intricate routines. While the program is somewhat lengthy and complex, we think the dedicated PC-2 owner will delight in experimenting with this utilitarian and instructive program.)

Enclosed is a program titled "SCRIP" which includes many short routines and mini-programs. An 8K Module and a printer are required. After CLOADing the program (of course, you will want to enter the program and then save it on tape before you begin experimenting), type (DEF) (M) and the Menu, which lists the options included, will be printed.

M MENU (Line 20)

=	PGM. NAME	F	PEEK ALL LIM EX0
A	ALARM		EX255
B	BANNERS FGHK	G	LOAD L$ 0-91
L	LABEL	H	C/COUNT65170-71
M	MENU	Z	POK
N	TIME	S	SIZES
J	CALENDAR		iF1 STATUS 2GLOBAL
SPC	DEFs		iF3OFF
D	SCRIP		i4time
X	COPIES		i5LF1 6LF-2
C	EDIT SCRIP	#	FREE 6
K	USED IN SCRIP	Y	COLORS 10
V	SHADOW 5000	*	BIORHYTHM 9110

The first sixteen routines/mini-programs are accessed by entering (DEF) and the appropriate Menu code such as (DEF) (M) to list the Menu, (DEF) (D) for the SCRIP program, (DEF) (B) for the Banners program, etc. The last seven routines (column 2) can be accessed by typing "G. " and the appropriate line number, i.e., G.5 for Status 2Global, G.9110 for Biorhythm, etc. For iF1 STATUS, you can also use Function Key 1; i4time, Function Key 4; i5LF1, Function Key 5; and 6LF-2, Function Key 6.

A ALARM (Line 64700)

The alarm routine is accessed by pressing (DEF) (A). The current time is displayed and then you are prompted for the alarm input in the form of: ALARM: HH.MMSS. After you input the alarm setting, the printer will write: ALARM @ MODAHH.MMSS and the display will show HH.MMSS until the alarm sounds. After the beep, the time will be printed and the display will return to the date/time setting, i.e. YYYY MODAHH.MMSS. (Note: After entering the program you may want to reset the time. This is done by typing: TIME = MODAHH.MMSS. In this case MO = month, DA = day, HH = hour, MM = minutes and SS = seconds. For more information about the time routine, see "N TIME.") While the alarm function is engaged, the computer cannot be used for other activities.

B BANNERS FGHK (Line 4400)

In order to use the "BANNERS" program, it is necessary to set up the RECIPE for each character or group of characters on one line. That is the user designs the content, the size (S), the number of characters per line (LEN), and direction of printing H/V) and must have previously determined the RECIPE factors of F G H and K. This can only be done by experimentation. To explain the recipe formula we'll use the following example.

RECIPE FORMULA (Line 64015)

S	LF	G	H	K	LEN
36	F-12	1	33	9	1L

First the size is listed (S) 36; the next four characters show the reverse line feed after each print (LF) F-12; then comes the line feed at the end of the 4th color change (G) 1; the total times the line is printed (H) 33; the line feed after the total lines are printed (K) 9; and finally the line length (LEN) 1L. The elapsed time to print a banner using this recipe was 57 minutes, 55.0 seconds or 7.02 Char/Sec (57*60 + 55)/33/15 = 7.020202) including 2 spaces 33 times. (If you try it, don't sit and watch — go for a jog.)

SIZES SUMMARY CHART (A Few of Thousands)

Vertical (Rotate 1)

S	LF	G	H	K	LEN	CONTENT	CHAR/SEC
6	F-10	0	5	10	5L	SCRIP	1.63
6	F-12	0	4	10	6L	SCRIP	1.37
7	F-12	0	2	12	5L	SCRIP	1.29
9	F- 4	1	9	4	1L	FINAL	3.59
13	F-13	1	13	12	3L	COLORS	2.60
14	F-22	0	33	22	5L	FINAL	2.53
14	F-22	0	33	22	5L	FOOT PRINT	2.37
23	F-22	1	33	23	3L	TRS-80	3.71
23	F-22	1	33	23	3L	ABCDEF	4.49
28	F-18	1	33	16	2L	ABCDEF	5.68
36	F-12	1	33	9	1L	ABCDEF	8.17
36	F-12	1	33	9	1L	ALL CHAR	7.21

Horizontal (Rotate 0)

GROUP 1

51	F-11	-1	9	17	1L	All except GROUP 2	10.00

GROUP 2

51	F	2	-1	9	27	1L	Only IJLNTUV WXZijl57!() < > and sq. root sign	8.37

The four characters, quotation mark(34), insert symbol (39), the Yen(92), and the Tilde(126), must be POKEd into the appropriate memory location, using the formula:

POKE(STATUS 3 + 7 + (L-1) * 18)),nnn

(nnn being the ASCII value for the character being POKEd)

For horizontal printing there appear to be at least two "families" of characters, relating to the manner in which the pen draws the character. For vertical printing, all of the printable ASCII characters seem to be in one "family."

PROCEDURE FOR MAKING A BANNER

1. Decide on the content. For our example we have chosen "SCRIP".
2. Select the size (S) and refer to Table 1 to get the F G H K values. For example if your choice were size (S) 23 with 3 characters per line (LEN) the F G H K variables would be -22, 1, 33, 23. The 33 (H) could be decreased to 5 or 9 or increased to as much as 45 for a satisfactory spacing pattern. It is desirable to have H = (multiples of 4) + 1 in order to have the pen finish in COLOR 0 (black).
3. For our first example we used the variables from TABLE 1 for size 6, 5L. (On our second example, we used the variables for size 9.) After turning on the PC-2 (with the program resident in memory) for the first example, we typed the following:

   F = -10, G = 0, H = 5, K = 10 (ENTER)

   These variables can be entered at the outset, or they can be INPUT when prompted later. If entered before initiating the program, they will appear in the DISPLAY at step 12 for a final check.
4. Press (DEF) (D).
5. The DISPLAY reads "OK? Y/N " If you are ready type (Y) (ENTER). If not ready, press (BREAK).
6. Prompt: SIZE? 2 OR (1-255)? Our response for the first example was 6; then press (ENTER)
7. Prompt: HORIZ. OR VERTICAL? H/V? Example 1, response: (V) (ENTER).
8. Prompt: NO. OF LINES? (91 MAX) For our first example we discovered we could input 5 characters per line. Two extra lines are required in the Banners program, therefore our response to the prompt for number of lines was 3, then (ENTER). (NOTE: The number of lines is always 2 more than the actual number. Following the last line of character input, the next to last line should contain a "FLAG" consisting of " + + "; this signals the program that content input has been completed. No input is entered on the last line. After typing the FLAG " + + " on the next to last line, press (ENTER). At the prompt for the last line hit (BREAK) — See Step 10.) In our second example, we discovered we should enter only one character per line for CSIZE 9, therefore our response to the prompt for number of lines was 7.
9. Prompt: NO. OF COPIES? Respond to the prompt for number of copies with 1 then (ENTER).
10. Prompt: TYPE (Preceded by a quick display of line count for each line, i.e., "1 OF 3," "2 OF 3," etc.)
    INPUT
    Line 1 OF 3, TYPE: SCRIP (ENTER)
    Line 2 OF 3, TYPE: + + (ENTER)
    Line 3 OF 3, TYPE: (BREAK)
11. Now press (DEF) (B)
12. The display reads "BANNERS:MISC. OR TRS? M/T" Type M then (ENTER).
13. The display should show the variables entered in step 3 which for our first example would be "-10 0 5 10." Check the variables for correctness, then press (ENTER).

```
F=-10, G=0, H=5, K=10
                  -10
                    0
                    5
                   10


            12120B.3219

S
C
R
I
P

            121208.3257
ELAPSED TIME
                0.003B0
S/C  1.52
USED:
CSIZE 6
DIRECTION U
# LINES 1
# COPIES 1
COLOR 0
1st LINE SCRIP
Nth LINE SCRIP
 6F-10 0 5 10 5L
```

Example 1

14. The display will read "F G H K." If the variables previously displayed were not correct then INPUT all variables again as shown in step 3, using the correct figures. If the variables were satisfactory, press (ENTER). The result is shown in our examples 1 and 2 and includes the (DEF) (K) data. (DEF) (K) lists the parameters selected for typing, for making Banners, and shows the "Recipe" in a compact form at the completion of all Banners. (See K USED IN SCRIP for more information.)
15. If for some reason you should decide to abort the RUN before completion by pressing (BREAK), let S=2, D$="H" to leave things tidy and start over again.

```
F=-4, G=1, H=9, K=4
                    -4
                     1
                     9
                     4
           121208.3627
```

```
           121208.3921
ELAPSED TIME
                  0.02539
S/C  3.86
USED:
CSIZE 9
DIRECTION U
# LINES 1
# COPIES 1
COLOR 0
1st LINE S
Nth LINE S
 9F-4 1 9 4 1L
```

Example 2

L LABEL (Line 65260)

This routine is set to print Mr. Brown's name, address and telephone number. You can substitute your own address data and have a quick print return address label for all your letters, packages, etc.

N TIME (Line 64900)

The time routine will display the current time setting in the form of YYYY MODAHH.MMSS, e.g., 1984 10309.1525. In order to leave this routine you must hit the (BREAK) key twice. You may find yourself in this routine when you turn the computer off for a short period and then turn it back on. Just hit the (BREAK) key twice to exit.

J CALENDAR (Line 6005)

Press (DEF) (J) to access this program. You will be prompted with: # MONTHS. Enter the number of months you want printed. For a year's calendar, enter 12. The next prompt requests the precise period to be covered with: START (MM.YYYY). To get our 1984 calendar, we entered: 01.1984.

```
 12 MONTHS
        122909.2637
JAN 1984
 S M T W T F S
 1 2 3 4 5 6 7
 8 91011121314
15161718192021
22232425262728
293031

FEB 1984
 S M T W T F S
 - - - 1 2 3 4
 5 6 7 8 91011
12131415161718
19202122232425
26272829

MAR 1984
 S M T W T F S
 - - - - 1 2 3
 4 5 6 7 8 910
11121314151617
18192021222324
25262728293031

APR 1984
 S M T W T F S
 1 2 3 4 5 6 7
 8 91011121314
15161718192021
22232425262728
2930

MAY 1984
 S M T W T F S
 - - 1 2 3 4 5
 6 7 8 9101112
13141516171819
20212223242526
2728293031

JUN 1984
 S M T W T F S
 - - - - - 1 2
 3 4 5 6 7 8 9
10111213141516
17181920212223
242526272829 30

JUL 1984
 S M T W T F S
 1 2 3 4 5 6 7
 8 91011121314
15161718192021
22232425262728
293031

AUG 1984
 S M T W T F S
 - - - 1 2 3 4
 5 6 7 8 91011
12131415161718
19202122232425
262728293031

SEP 1984
 S M T W T F S
 - - - - - - 1
 2 3 4 5 6 7 8
 9101112131415
16171819202122
23242526272829
30
OCT 1984
 S M T W T F S
 - 1 2 3 4 5 6
 7 8 9101112 13
14151617181920
21222324252627
28293031

NOV 1984
 S M T W T F S
 - - - - 1 2 3
 4 5 6 7 8 910
11121314151617
18192021222324
252627282930

DEC 1984
 S M T W T F S
 - - - - - - 1
 2 3 4 5 6 7 8
 9101112131415
16171819202122
23242526272829
3031
 12 MO.  122909.322
6
ELAPSED TIME
                  0.05490
SEC./MO.     29.0
```

Example 3

SPC DEFs (Line 65277)

Access this routine by pressing (DEF) (SPACE BAR). It will produce a list of all of the DEFined labels as well as many other symbols and even numerals that can be used for addresses for GOTOs, GOSUBs, and just plain markers for significant addresses. The line numbers are also listed.

D SCRIP (Line 62010)

SCRIP converts the PC-2 into a mini-typewriter, to type memos, explanatory notes, labels, etc. during program development and/or execution, and to experiment with an apparently new area of pattern development by the use of sizes larger that the usual CSIZE9. The program is designed for up to 91 lines with up to 18 characters per line; however, the user can also select any size from 1 to 255. (For an illustration of this facility, run the Sizes Demonstration program — (DEF) (S).)

Thanks to Mr. Norlin Rober, Master Sleuth, the use of &79F (Dec 31220) makes it easy to develop some interesting and unusual patterns with these larger size characters.

PROCEDURE FOR USING SCRIP

1. To initiate the SCRIP program, press (DEF) (D).
2. The display shows: OK?Y? To proceed, type "Y" and then (ENTER).
3. The display then reads: SIZE? 2 OR (1-255) ? Enter the character size you want printed. If you do not enter a number and just press (ENTER), the program will default to size 2.
4. The next prompt is: HORIZ. OR VERTICAL? H/V Enter H or V. The default for (ENTER) is horizontal (H).
5. Display will prompt: NO OF LINES ? (91 MAX) Enter the number of lines desired and press (ENTER); the default entry for (ENTER) is 1.
6. Prompt: NO. OF COPIES ? Enter the number of copies desired and press (ENTER). If you do not enter a number and just press (ENTER) the default entry is 1.

At this point you are ready to begin typing. A prompt appears for the first line entry of up to 18 characters. After the brief flash of "1 OF 3," you will see the TYPE: prompt. For our example 4 we used the default size (2), the default direction (H), 3 lines, and 1 copy. For example 5 we varied the size only, using 5 to show how the program will accept up to 18 characters per line but the output will wrap around if all the characters for a line cannot be physically printed on a single line. After the last of N lines is typed and ENTERed the requested number of copies is automatically printed.

```
THIS IS THE SCRIP
PGM. UP TO 18 C/L
123456789012345678
```

Example 4

If additional copies are needed later (before CLEARing or reDIMensioning by this or any other program) just press (DEF) (X) and follow the prompts. After the number of extra copies is ENTERed, press any character, then press (ENTER) once more and the extra copies will be delivered.

Although the SCRIP program is designed for up to 91 lines (required to test that all of the printable ASCII characters could be perfectly printed at the largest possible size), the

```
THIS IS
 THE SC
RIP
PGM. UP
 TO 18
C/L
1234567
8901234
5678
```

Example 5

"SCRIP" DIMension can be reduced to 50 lines, or any desired number, at a savings of 18 bytes per line by the following changes: (using 50 as an example)

In line 62010 change DIM L$(91) to DIM L$(50)
then in line 62070 change (91 MAX) to (59 MAX)

VARIABLES IN THE SCRIP PROGRAM

C	Number of Copies selected (Default = 1)
D	Copies counter
E	Used in Def C (Change or edit a line)
F G H K	Used in DEF B (Banners), DEF K, etc. For "RECIPE" only, use G.64015 or GOTO 64015
L	Number of Lines selected (Default = 1)
M	Lines Counter
N	Line input pointer
S	Size selected (1 to 255; Default = 2) For sizes other than 1-9 use POKE &79F4 Decimal 31220. (CAUTION!! LOOK FIRST!! Then POKE)
L$(91)*18	DIMs for up to 92 lines, with up to 18 characters per line. User may increase or decrease the number of lines depending on available memory. Each line requires 18 locations (1663 for 92 lines). A total of 92 lines are necessary to check the full ASCII list for the large sizes of printing (including housekeeping) for edit, additional copies, etc. For only 50 lines (a saving of 738 locations) make the following changes: Line 62010 L$(50) Instead of L$(91) Line 62070 L>50 Instead of L>91 and 50MAX Instead of 91MAX
D$	Direction selected (H/V in D$) Default = H

AA B$ Z$ B I J Q T W X Z Also used in Banners, etc.
NOTE: DON'T TRY TO CHANGE COLORS BY POKEing.

X COPIES (Line 62080)

This is a subroutine to be used with SCRIP or Banners. You are prompted for the number of copies desired in the

SCRIP program but extra copies can be called for until the memory has been CLEARed by this or some other program. After your entry has been printed, press (DEF) (X) and enter the number of extra copies you want. You will be prompted with the TYPE: prompt. Hit any key and (ENTER).

It should be noted (Line 62010) that CLEAR is not used before DIMensioning. Use is made of CALL&D091 which permits the retention of the fixed Variables (A-Z) for use in some of the other related programs such as DEF K, the Banners, the RECIPE, etc.

C EDIT SCRIP (Line 62500)

After you have entered your SCRIP lines or after you have run your entry once, you may want to change the text. This subroutine lets you change your entry but the entire line must be changed.

1. Press (DEF) (C)
2. The display will prompt: WHICH LINE NO.? If you want to change line 5, then press (5) and (ENTER).
3. The display will show the current line content such as: 5 is GAME. Press (ENTER)
4. Display will prompt: CHANGE TO . . . . Type your response and (ENTER).
5. To print your new SCRIP entry with the change you can use (DEF) (X) and print 1 or more copies. Although only entire lines can be changed with this routine, a means is available to POKE certain characters (quotation mark, the tilde, the insert symbol, and the Yen sign) into the appropriate memory location. You can use the Z POK routine; just type (DEF) (Z) to access. The four characters, quotation mark(34), insert symbol(39), the Yen(92), and the tilde(126), are POKEd into the appropriate memory location by using the formula:
   POKE(STATUS 3 + 7 + (L-1)*18)),nnn (nnn = ASCII value)

K USED IN SCRIP (Line 64000)

Lists the parameters selected for typing, for making Banners, and shows the "RECIPE" in a compact form at the completion of all Banners. In experimenting with the various sizes, characters per line (LEN), etc., this record was found to be extremely valuable. Entering (DEF) (K) will result in a printout listing the CSIZE, direction, number of lines, number of copies, the color pen used, the content of the first line, the content of the last line, and the Recipe Formula used in the last SCRIP or Banners entry.

G LOAD L$ 0-91 (ALL ASCII CHARACTERS)

This program produces a sample run of all characters. It was found that many characters could be printed at Size 36 <Vertical> and at Size 51 <Horizontal>. However, Size 51 <H> required two groups, with different "RECIPES." This routine, accessed by typing (DEF) (G), can only be used with the Size 36 Vertical sample. In order to make certain that all the ASCII characters could be printed without distortion, this routine LOADS all 91 into L$(0) to L$(90), the last one being for the FLAG " + + ".

Procedure For Size 36 Vertical

1. Set the F G H K values: F = -12 G = 1 H = 1 K = 9 We chose to set H at 1 to save time and paper. At this setting it will take 12 minutes.
2. Press (DEF) (D), Type (Y), then (ENTER).
3. Type 36 (ENTER) at the size prompt.
4. Type V (ENTER) at the H/V prompt.
5. Then Press (BREAK)
6. Press (DEF) (G). This command will LOAD the ASCII list into L$(0) to L$(90), including the FLAG " + + " (Program line 8030).
7. Press (DEF) (B)
8. Type M (ENTER) at the M/T prompt.
9. The Display should show "-12 1 1 9" (the values set in step 1 above). Press (ENTER)
10. Press (ENTER) again and the ASCII list will be delivered in about 12 minutes.

Procedure For Size 51 Horizontal

The characters must be individually entered, line by line, in two groups.

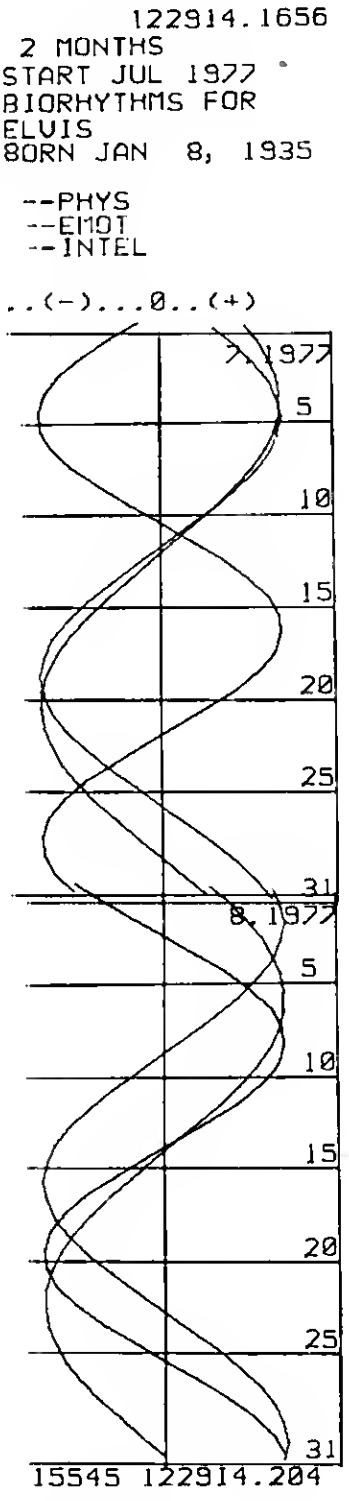
1. For Group 1 (all characters except Group 2) the F G H K values are: F = -11 G = 1 H = 1 K = 17. The values for Group 2 (IJLNTUVWXZ57!() < > and the square root sign) are: F = 2 G = -1 H = 1 K = 17. (If you want shadow printing then H = 5 or more in each case.)
2. Press (DEF) (D), Type (Y), then (ENTER).
3. Type 51 (ENTER) at the size prompt.
4. Type H (ENTER) at the H/V prompt.
5. Number of lines for Group 1: 71
   Number of lines for Group 2: 23
   The last two lines in each case are:
   Line 70/22: + +
   Line 71/23: (BREAK)
6. When entering Group 1, use dummies for the quotation mark(34), insert symbol(39), yen(92), and the tilde(126) which must be POKEd to the dummies. After the (BREAK) on line 71 for Group 1, (DEF) (Z) and POKE using the formula:
   POKE(STA.3 + 7 + (L-1)*18)),nnn (nnn = the ASCII value of the character being POKEd)
7. Press (DEF) (B) and respond M (ENTER) to the T/M query. If the FGHK values are correct press (ENTER) again and you will get a printout of all characters at Size 51.

S SIZES (Line 65220)

This program provides a demonstration of the sizes from 1 to 255 and shows the number of characters per line that are possible at each size. The size number is POKEd into Memory Location &79F4 (Decimal 31220) for sizes larger than CSIZE9. The resulting tape requires 27 minutes +/-.

* BIORHYTHM (Line 9110)

Accessed by typing: G.9110 or G." * ", this program will produce a biorhythm chart. After typing G.9110 (ENTER), you will see the prompt: START(MM.YYYY). Enter the date of the month on which you want the chart to begin. You are then asked to enter the name of the person for which the chart is being made. Next you are asked to enter that person's birth date (MMDD.YYYY) and finally the number of months to be covered by the chart. Mr. Brown included a sample chart for Elvis.



Example 6

DEFined LABELS, MARKERS FOR GOTO AND GOSUB ADDRESSED

In addition to the usual DEF LABELS (ASDFGHJKLZX-CVBNM= and the space bar), most if not all of the other characters, including numerals and those not available from the keyboard can be used as GOTO and GOSUB addresses in the program as well as manually from the keyboard, if they are used as STRINGs. For example, the use in lines 4000 "@ . . . " and 4420 GOTO "@ . . . " was very helpful as the entire program had already used up all of the orthodox DEF labels. Some memory can also be saved if the line numbers are above 3 digits and referred to often.

PAPER AND PENS

Either the small rolls (26-3606) or the larger Calculator Paper (65-710) can be used. When printing banners with the larger sizes with the reverse line feed of LF-6 to LF-24, the roll of paper should be put on an external stand with plenty of slack between the paper and the printer. The testing of these exercises probably provided the ultimate test of the ruggedness of the printing mechanism of the TRS Printer-Interface. The size 36 Banner test for all 91 characters took almost 6 hours of continuous printing and during that time the pens probably printed almost 1/4 mile of inked lines (a guesstimate).

MEMORY REQUIREMENTS

Scrip and Error Routine	641		
Shadow Printing	87		
K Parameters Used	183		
Subtotal		911	
Banners Misc or TRS? M/T	626		
Elapsed and Average Time	135		
L$ (90-91) Load and Print	252		
Subtotal		1013	
Total SCRIP et al			1924
Sizes Demonstration	1483		
Colors	78		
ARUN and time LCD	57		
Characters per line DEF H	134		
DEFs DEF " "	117		
Subtotal		1869	
Calendar	1438		
Biorhythms	1550		
Program Name, Menu, Etc.	1576		
Subtotal		4564	
TOTAL (Status 2)			8357

AND EVEN MORE

There is more to explore in this program. DEF H counts the number of characters per line and lists number/ per line and cumulative; DEF F is a PEEK routine which prompts: ALL,LIM,EXO,EX255; DEF = prints the Program name and creation data; and DEF V is a shadow print routine which functions as a subroutine of the Banner program. And that's not all. DEF R produces a GOTO prompt which provides quick access to other parts of the program such as: DEF R 6 provides the number of free bytes; DEF R Y produces a test of the color pens; and DEF R 5 produces a status printout.

(Editor's Note: The program requires all the available memory space. You may want to NEW0 before you begin entering the program. Program line annotations were provided by the author. A very large THANK YOU and tip of our hat to Mr. Brown for his contribution.)

SCRIP PROGRAM

```
1 ARUN
     : LF -5
     : GOTO "1"
2 "+" L=(W-INT W)*1E4
     : M=INT (W/1ØØ)
     : N=INT W-(INT ((INT W)/1ØØ))*1ØØ
     : RETURN
3 "E" BEEP 7
     : PAUSE "OOPS ERROR1"
     : END
4 "O" LPRINT "GLOBAL";TIME
     : GOSUB 14
     : GOSUB 2Ø
     : GOSUB 5
     : ENO
5 "P" TEXT
     : USING
     : CSIZE 2
     : COLOR 3
     : LPRINT "STATUS"
     : COLOR Ø
     : LPRINT "19B3";TIME
     : FOR I=Ø TO 4
     : LPRINT I;STATUS I
     : NEXT I
6 "#" ON ERROR GOTO 3
     : USING
     : LPRINT "FREE";STATUS 3-STATUS 2
     : RETURN
7 "T" Z=DMS ((DEG TIME )-T)
     : LPRINT TIME
     : LPRINT USING "###.#####";"ELAPSED TIME",Z
     : USING
     : RETURN
8 "W" Z$=STR$ Z
     : V=36ØØ*(VAL LEFT$ (Z$,2))+6Ø*(VAL MID$
     (Z$,3,2))+VAL MID$ (Z$,5,3)/1Ø
     : RETURN
9 "U" Q$(Ø)="JANFEBMARAPRMAYJUNJULAUGSEPOCTNOVOEC"
     : RETURN
1Ø "Y" COLOR Ø
     : LPRINT "TEST"
     : LPRINT "BLACK";
     : COLOR 1
     : LPRINT "BLUE";
     : COLOR 2
     : LPRINT "GREEN";
     : GOLOR 3
     : LPRINT "REO"
     : COLOR Ø
     : ENO
14 "=" COLOR 3
     : USING
     : LPRINT "SCRIP";
     : COLOR Ø
     : LPRINT " FT.W 7/29/83"
     : LPRINT "19B3";TIME
     : LPRINT "39A42A 2-1B6RES193
18 LPRINT "ORIG
     : 16B5.835B"
     : LPRINT "NOW ";STATUS Ø;STATUS 1
     : LPRINT "3Ø823.14533"; PEEK 3Ø823; PEEK 3ØB24
     : GOSUB 97
19 ON ERROR GOTO 3
     : LPRINT "CR2"
     : RETURN
2Ø "M" COLOR 3
     : LPRINT "MENU"
     : COLOR Ø
     : ON ERROR GOTO 3
25 LPRINT "= PGM. NAME","A ALARM"
     : LPRINT "B BANNERS FGHK"
     : LPRINT "L LABEL","M MENU
3Ø LPRINT "N TIME"
     : LPRINT "J CALENOAR","SPC DEFs
4Ø LPRINT "O SCRIP","X COPIES"
     : LPRINT "C EOIT SCRIP", "K USEO IN SCRIP"
     : LPRINT "V SHADOW 5ØØØ
5Ø LPRINT "F PEEK ALL LIM EXØ  EX255"
     : LPRINT "G LOAO L$ Ø-91"
     : LPRINT "H C/COUNT 6527Ø-71","Z POK
6Ø LPRINT "S SIZES","iF1 STATUS 2GLOBAL"
     : LPRINT "iF3OFF","i4time"
     : LPRINT "i5LF1 6LF-2
85 LPRINT "# FREE 6","Y COLORS 1Ø"
     : LPRINT "* BIORHYTHM 911Ø    iiiF1"
     : RETURN
97 "R" R$=STR$(PEEK &C443) + STR$(PEEK &C5BD)
     : IF R$="56129" LPRINT R$;" REV. AØ1"
     : RETURN
98 IF R$="59129" LPRINT R$;" REV. AØ3"
     : RETURN
99 IF R$="5974" LPRINT R$;" REV. AØ4"
     : RETURN
3ØØØ "%"  FOR AA=1 TO 26
     : IF @(AA)<>7Ø7 LPRINT CHR$ (AA+64);@(AA)
3Ø1Ø NEXT AA
     : BEEP 5
     : END
31ØØ "3" FOR AA=1 TO 26
     : @(AA)=7Ø7
     : NEXT AA
     : END
32ØØ "?" FOR B=91 TO Ø94
     : LPRINT "";B
     : NEXT B
     : ENO
4ØØØ "@" WAIT
     : PRINT F;G;H;K
     : INPUT "F,G,H,K";F,G,H,K
4Ø1Ø T=DEG TIME
     : LPRINT TIME
     : LF 1
4Ø5Ø L=1
     : Q=Ø
     : L$=L$(Ø)
     : W=Ø
41ØØ L$(Ø)=L$(Q)
     : W=W+LEN L$(Ø)
     : PAUSE L$(Ø);L;M;N
     : IF L$(Ø)="++" THEN 413Ø
412Ø GOSUB "V"
     : Q=Q+1
     : GOTO 41ØØ
413Ø GOSUB 7
     : GOSUB 8
     : L$(Ø)=L$
     : X=V/(W-2)/H
     : LPRINT USING "###.##";"S/C";X
     : USING
     : GOTO "K
44ØØ "B" INPUT "BANNERS
     : MISC.OR TRS? M/T";B$
     : IF B$="T" THEN "Q
441Ø IF B$<>"M" THEN "B
442Ø GOTO "@
45ØØ "Q" F=-22
     : G=1
     : H=13
     : K=23
     : INPUT "TRS F,G,H,K";F,G,H.K
451Ø S=23
     : O$="V"
     : C=1
     : T=DEG TIME
     : LPRINT TIME
     : W=Ø
     : LF 2
     : L$="TRS
```

```
4550 RESTORE 4650
4600 READ L$(0)
     : W=W+LEN L$(0)
     : IF L$(0)="++" THEN 4630
4620 GOSUB 5000
     : GOTO 4600
4630 GOSUB 7
     : GOSUB 8
     : X=V/(W-2)/H
     : LPRINT USING "###.##";"S/C";X
     : USING
4640 CSIZE 1
     : LPRINT "AIN^T THAT SUMP^N!"
     : CSIZE 2
     : LF 3
     : GOTO "K
4650 DATA "TRS","-80"," PC","-2 ","TAN","OY ","RAD",
     "IO ","SHA","CK ","++
5000 "V" J=0
     : USING
5010 FOR I=0 TO 3
     : D=0
     : COLOR I
     : GOSUB 62110
     : LF F
     : J=J+1
     : PAUSE H;J
     : IF J<H NEXT I
     : LF G
     : GOTO 5010
5020 LF K
     : RETURN
6005 "J" COLDR 0
     : CALL &D091
     : DIM A$(43)*2,Q$(0)*36
     : INPUT "# MONTHS ";X
     : GOSUB 9
6010 INPUT "START (MM.YYYY)";F
     : USING
     : LPRINT X;" MONTHS
6011 T=DEG TIME
     : LPRINT TIME
     : A=INT F
     : B=INT ((F-A)*1E4)
     : Y=0
6015 Y=Y+1
     : IF Y<X+1 RESTORE 6022
     : GOTO 6019
6016 LPRINT X;" MO.";
     : GOSUB 7
6017 LPRINT USING "###.#";"SEC./MO.  ";
     (60*INT(Z*100)+100*(100*Z-INT(Z*100)))/X
     : USING
     : END
6019 COLOR 3
     : LPRINT MID$ (Q$(0),(A*3-2),3);8
     : COLOR 0
     : IF B<1700 OR A+B<1703 OR B+A/10>2200.2
     THEN 6400
6021 FOR I=12 TO 42
     : READ A$(I)
     : NEXT I
     : GOTO 6045
6022 DATA " 1"," 2"," 3"," 4"," 5"," 6"," 7"," 8",
     " 9","10","11","12","13","14
6023 DATA "15","16","17","18","19","20","21","22",
     "23","24","25","26","27","28
6024 DATA "29","30","31
6045 E=621048
     : C=365.25
     : IF A<=2 LET D=INT(30.6*(A+13) +
     INT(C*(B-1))-E
     : GOTO 6055
6050 D=INT(30.6*(A+1))+INT(B*C)-E
6055 IF D>146097 LET D=O-1
6060 IF O<=73047 LET D=D+1
     : IF D<=36522 LET O=D+1
6075 D=D/7
     : D=INT(7*(D-INT D)+1.49)
     : IF A=4 DR A=6 DR A=9 OR A=11 LET C=30
     : GOTO 6105
6080 IF A=1 OR A=3 OR A=5 OR A=7 OR A=8 OR
     A=10 OR A=12 LET C=31
     : GOTO 6105
6090 C=28
     : IF (B/4=INT(B/4))+(B/1E2<>INT(B/1E2))=2
     LET C=29
6095 IF B=2E3 LET C=29
6105 FOR E=1 TO D-1
     : A$(E+5)=" -"
     : NEXT E
6110 FOR E=1 TO C
     : A$(O+E+4)=A$(E+11)
     : NEXT E
     : FOR E=D+C+5 TO 42
     : A$(E)=""
     : NEXT E
6125 LPRINT " S M T W T F S"
     : LPRINT A$(6);A$(7);A$(8);A$(9);A$(10);
     A$(11);A$(12)
6130 LPRINT A$(13);A$(14);A$(15);A$(16);A$(17);
     A$(18);A$(19)
6135 LPRINT A$(20);A$(21);A$(22);A$(23);A$(24);
     A$(25);A$(26)
6140 LPRINT A$(27);A$(28);A$(29);A$(30);A$(31);
     A$(32);A$(33)
6145 LPRINT A$(34);A$(35);A$(36);A$(37);A$(38);
     A$(39);A$(40)
     : LPRINT A$(41);A$(42)
6153 A=A+1
     : IF A<=12 GOTO 6015
6155 A=1
     : B=B+1
     : GOTO 6015
6400 BEEP 7
     : LPRINT TIME ;" EARLI- EST VALID MONTH 3/1700;
     LAST 2/2200"
     : GOTO 6005
8000 "G" FOR I=65 TO 90
     : L$(I-65)=CHR$ I
     : NEXT I
     : FOR I=97 TO 122
     : L$(I-71)=CHR$ I
     : NEXT I
8010 FOR I=48 TO 57
     : L$(I+4)=CHR$ I
     : NEXT I
     : FOR I=32 TO 47
     : L$(I+30)=CHR$ I
     : NEXT I
8020 FOR I=58 TO 64
     : L$(I+20)=CHR$ I
     : NEXT I
     : FOR I=91 TO 94
     : L$(I-6)=CHR$ I
     : NEXT I
8030 L$(89)=CHR$ (126)
     : L$(90)="++"
     : ENO
8100 "PI " USING
     : FOR I=0 TO 91
     : LPRINT I;" ";L$(I)
     : NEXT I
     : ENO
9110 "*" USING
     : CLEAR
     : DIM A$(0)*18,Q$(0)*36
     : INPUT "START (MM.YYYY";Z
     : GOSUB 9
     : M=INT Z
     : L=(Z-M)*1E4
9120 G=L
```

```
     : H=M
     : TEXT
     : COLOR Ø
     : GOSUB 97ØØ
     : GOSUB 96ØØ
     : O=A
     : INPUT "NAME? ";A$(Ø)
9125 INPUT "BORN (MMOO.YYYY";L$
     : W=VAL L$
913Ø GOSUB 2
     : INPUT "# MONTHS? ";U
     : LPRINT TIME
     : LPRINT U;" MONTHS
914Ø LPRINT "START ";USING "#####";MID$ (Q$(Ø),
     (H*3-2),3);G
     : LPRINT "BIORHYTHMS FOR"
     : COLOR 3
     : LPRINT A$(Ø)
9145 COLOR Ø
915Ø LPRINT "BORN ";MIO$ (Q$(Ø),(M*3-2),3);
     USING "###";N;",";USING "#####";L
     : GOSUB 96ØØ
     : P=A
     : A=O-P
9155 O=Ø
     : P=Ø
     : LF 1
     : USING
918Ø COLOR 3
     : LPRINT " --PHYS"
     : COLOR 1
     : LPRINT " --EMOT"
     : COLOR 2
     : LPRINT " --INTEL"
     : LF 1
     : COLOR Ø
     : LPRINT "..(-)...Ø..(+)
919Ø LF 1
     : LPRINT Z
     : LF -2
92Ø5 GRAPH
92Ø8 GLCURSOR (1ØØ,Ø)
     : SORGN
     : Y=I*-12.5
     : LINE (-1ØØ,Ø)-(115,Ø)
     : LINE (Ø,Ø)-(Ø,Y)
     : LINE (115,Y)-(115,Ø)
924Ø FOR Q=5 TO 3Ø STEP 5
     : R=Q
     : IF Q=3Ø LET R=I
925Ø Y=R*-12.5
     : LINE (-9Ø,Y)-(115,Y)
     : X=8Ø
     : S=Y+5
     : LINE (115,S)-(X,S),9
     : LPRINT USING;R
     : NEXT Q
933Ø B=INT (A/23)
     : B=A-(23*B)
     : C=INT (A/28)
     : C=A-(28*C)
     : D=INT (A/33)
     : O=A-(33*O)
     : FOR J=1 TO 3
942Ø COLOR J
     : E=Ø
     : FOR Y=Ø TO I
     : IF J=3 LET X=SIN ((B+Y)/23*36Ø)*8Ø
943Ø IF J=1 LET X=SIN ((C+Y)/28*36Ø)*8Ø
944Ø IF J=2 LET X=SIN ((O+Y)/33*36Ø)*8Ø
947Ø S=Y*-12.5
     : F=Ø
     : IF E=Ø LET F=9
     : E=1
9517 LINE (O,P)-(X,S),F
     : O=X
     : P=S
     : NEXT Y
     : NEXT J
     : TEXT
     : LF 1
     : COLOR Ø
     : T=T+1
     : IF U>T THEN 1ØØ1Ø
952Ø USING
     : LPRINT A;TIME
     : ENO
96ØØ IF M-3>=Ø LET M=M+1
     : GOTO 962Ø
961Ø L=L-1
     : M=13+M
962Ø A=INT (365.25*L)+INT (3Ø.6*M)+N
     : A=A-INT (L/1ØØ)+INT (L/4ØØ)
     : RETURN
97ØØ IF M=2 GOTO 979Ø
974Ø IF M=4 OR M=6 OR M=9 OR M=11 GOTO 977Ø
975Ø I=31
     : GOTO 99ØØ
977Ø I=3Ø
     : GOTO 99ØØ
979Ø K=INT (L/4)
     : K=L-K*4
98ØØ IF K=Ø GOTO 984Ø
982Ø I=28
     : GOTO 99ØØ
984Ø K=INT (L-1ØØ)
     : K=L-K*1ØØ
     : IF K=Ø GOTO 985Ø
9847 GOTO 989Ø
985Ø K=INT (L/4ØØ)
     : K=L-K*4ØØ
     : IF K=Ø GOTO 989Ø
987Ø GOTO 982Ø
989Ø I=29
99ØØ RETURN
1ØØ1Ø V=Z+1
     : IF V>13 LET V=V-11.9999
1ØØ15 Z=V
     : M=INT Z
     : L=(Z-M)*1E4
     : G=L
     : H=M
     : TEXT
     : COLOR Ø
     : IF H>12 LET H=1
     : G=G+1
     : L=G
     : M=H
1ØØ6Ø TEXT
     : COLOR Ø
     : N=Ø
     : GOSUB 97ØØ
     : GOSUB 96ØØ
     : O=A
     : GOSUB 2
     : LPRINT USING "###.####";V
     : LF -2
1Ø11Ø GOSUB 96ØØ
     : P=A
     : A=O-P
     : O=Ø
     : P=Ø
     : USING
     : GOTO 92Ø5
61ØØØ "F" INPUT "ALL,LIM,EXØ,EX255";P$
61Ø1Ø IF P$="ALL" LET P=.1
     : Q=P
     : GOTO 611ØØ
61Ø2Ø IF P$="LIM" LET P=Ø
     : Q=255
     : GOTO 611ØØ
61Ø3Ø IF P$="EXØ" LET P=Ø
```

```
     : Q=P
     : GOTO 611ØØ
61Ø4Ø IF P$="EX255" LET P=255
     : Q=P
     : GOTO 611ØØ
61Ø45 BEEP 3
     : GOTO "F
611ØØ PAUSE P$
     : ON ERROR GOTO 6116Ø
     : USING
     : D=32768
     : INPUT "FROM ";A
     : INPUT "TO ";B
     : LPRINT TIME
     : LPRINT A;B;P$
6113Ø FOR I=A-D TO B-D
     : J=PEEK (I+D)
     : IF J=P OR J=Q NEXT I
6115Ø LPRINT I+D;J;" ";CHR$ J
     : NEXT I
6116Ø LPRINT P$;TIME
     : END
614ØØ "Z" USING
     : INPUT "POKE FROM ";A
     : INPUT "TO ";B
     : INPUT "VAL ";C
     : LPRINT A;B;"POK"
     : LPRINT "C=";C;TIME
6143Ø FOR I=A TO B
     : POKE I,C
     : NEXT I
     : LPRINT "PKD";TIME
     : BEEP 3
     : END
62ØØØ "D" INPUT "OK?Y/N";D$
     : IF D$<>"Y" END
62Ø1Ø "SQR " CALL &DØ91
     : DIM L$(91)*18
     : USING
     : S=2
     : INPUT "SIZE? 2 OR (1-255)?";S
62Ø2Ø IF S<1 OR S>255 BEEP 4
     : GOTO "SQR
62Ø3Ø D$="H"
     : INPUT "HORIZ. OR VERTICAL? H/V?";D$
     : IF D$="V" THEN 62Ø7Ø
62Ø5Ø IF D$<>"H" BEEP 4
     : GOTO 62Ø3Ø
62Ø7Ø TEXT
     : L=1
     : M=Ø
     : INPUT "NO. OF LINES? (91 MAX)";L
     : IF L<1 OR L>91 BEEP 4
     : GOTO 62Ø7Ø
62Ø8Ø "X" C=1
     : D=Ø
     : INPUT "NO. OF COPIES? ";C
62Ø9Ø WAIT 1Ø
     : PRINT M+1;" OF";L
     : BEEP 2
     : INPUT "TYPE :";L$(M)
     : M=M+1
     : IF M>L-1 THEN 6211Ø
621ØØ GOTO 62Ø9Ø
6211Ø IF D$="V" GRAPH
     : ROTATE 1
6212Ø M=Ø
     : POKE 31222,S
     : FOR N=1 TO L
     : LPRINT L$(M)
     : M=M+1
     : NEXT N
     : D=D+1
     : IF C=D THEN 6214Ø
6213Ø LPRINT " "
     : GOTO 6211Ø
6214Ø CSIZE 2
     : GRAPH
     : ROTATE Ø
     : TEXT
     : LPRINT
     : ON ERROR GOTO 3
     : RETURN
625ØØ "C" INPUT "WHICH LINE NO.? ";E
     : IF E<1 OR E>L BEEP 3
     : GOTO "C
6252Ø WAIT
     : PRINT E;" is ";L$(E-1)
     : INPUT "CHANGE TO ";L$(E-1)
     : PRINT L$(E-1)
     : END
64ØØØ "K" USING
     : LPRINT "USED:"
     : LPRINT "CSIZE";S
     : LPRINT "DIRECTION ";D$
     : LPRINT "# LINES";L
64Ø1Ø LPRINT "# COPIES";C
     : LPRINT "COLOR";PEEK 31219
     : LPRINT "1st LINE ";L$(Ø)
     : LPRINT "Nth LINE ";
64Ø15 ON ERROR GOTO 3
     : LPRINT L$(M-1)
     : LPRINT S;"F";F;G;H;K;LEN L$;"L"
     : RETURN
647ØØ "A" LPRINT TIME
     : INPUT "ALARM HH.MMSS:";U
     : PAUSE U
     : LF 1
     : CSIZE 3
     : T=INT (TIME /1ØØ)*1ØØ+U
     : LPRINT "ALARM @ ",T
6482Ø IF TIME =T LPRINT T
     : BEEP 5
     : CSIZE 2
     : GOTO "I
6483Ø GOTO 6482Ø
649ØØ "N" LPRINT "1983";TIME
652ØØ "I" PAUSE USING "#######.####";"1983";TIME
     : GOTO "I
6522Ø "S" TEXT
     : CLEAR
     : DIM L$(2)*18
     : CSIZE 2
     : COLOR Ø
     : T=DEG TIME
     : LPRINT TIME
     : CSIZE 3
     : LPRINT ""
     : LPRINT "A DEMONSTRA-TION OF
65221 L$(Ø)="SIZES"
     : S=14
     : D$="V"
     : L=2
     : C=1
     : F=-22
     : G=Ø
     : H=9
     : K=23
     : GOSUB "V
65222 LPRINT "FOR THE PC-2."
     : LPRINT "THE PRINTER MANUAL ADMITS(PAGE 63) TO
     ONLY 9 SIZES
65223 CSIZE 1
     : LPRINT "CSIZE 1"
     : LF 1
     : CSIZE 2
     : LPRINT TAB 8;"TO"
     : LF 3
     : CSIZE 9
     : LPRINT "NINE"
     : CSIZE 2
     : LF -2
```

```
65224 LPRINT "BUT BY POKEing A  BIT ONE REALLY
     CAN PRINT UP TO SIZE 255 AND SOME ";
65226 LPRINT "OTHER QUIRKS.
6523Ø COLOR 3
     : LPRINT "  9 SIZES AND","CHAR/LINE @ SIZE"
     : COLOR Ø
     : LPRINT
     : LPRINT "CSIZE          C/L"
     : CSIZE 1
65232 LPRINT "1 34567891";
65233 LPRINT TAB 26;"7893123 36"
     : CSIZE 2
     : LPRINT
     : LPRINT "2 3456789112345 18
65234 LPRINT
     : CSIZE 3
     : LPRINT "3 3456789 12"
     : CSIZE 4
     : LPRINT "4 3456789
65235 CSIZE 5
     : LPRINT "5 34567"
     : CSIZE 6
     : LPRINT "6 3456"
     : CSIZE 7
     : LPRINT "7 345
65236 CSIZE 8
     : LPRINT "8234"
     : CSIZE 9
     : LPRINT "9234"
     : CSIZE 2
     : LF -3
     : LPRINT "SIZES 1Ø-12","3 CHAR/LINE
65241 LF 3
     : Y=31222Ø
     : POKE Y,1Ø
     : LPRINT "1Ø3"
     : CSIZE 2
     : LF 2
     : POKE Y,12
     : LPRINT "123"
     : CSIZE 2
     : LF -3
65243 LPRINT "SIZES 13-18","2 CHAR/LINE"
     : LF 4
     : POKE Y,13
     : LPRINT "13"
     : CSIZE 2
     : POKE Y,18
65245 LPRINT "18"
     : CSIZE 2
     : LF -6
     : LPRINT "SIZES 19-55","1 CHAR/LINE; ABOVE
65246 LPRINT "55 MAY BE MISSHAPEN."
     : LF 6
     : POKE Y,19
     : LPRINT "1955"
     : POKE Y,39
65247 LPRINT "39"
     : LF 4
     : POKE Y,55
     : LPRINT "55"
     : LF 3
     : CSIZE 2
6525Ø LPRINT "AND NOW SIZE 55 @"
     : LF 11
     : POKE Y,55
     : LPRINT "@"
     : CSIZE 28
     : LF -3
65251 LPRINT "AND EVEN SIZE 255 FOR A PERIOD."
     : POKE Y,255
     : LPRINT "."
     : CSIZE 2
     : LF -18
65252 LPRINT TAB 9;"."
     : LF -1
65253 LPRINT "  OUCH ! ."
     : LF 7
     : LPRINT "LOOK WHAT THEY^VE DONE TO MY  *"
     : LPRINT "  * ASTERISK *
65254 LF 2
     : CSIZE 6
     : LPRINT "   *"
     : LF -1
65255 L$(Ø)="**********"
     : L$(1)=L$(Ø)
     : S=43
     : D$="V"
     : L=2
     : C=3
     : FOR I=Ø TO 3
65256 D=Ø
     : COLOR I
     : GOSUB 6211Ø
     : NEXT I
65259 LF 9
     : GOSUB "K"
     : COLOR Ø
     : LPRINT
     : LPRINT "YOU TRY 55 TO 255. USE MY
     SCRIP PROGRAM.
6526Ø "/" LPRINT "MARION F. BROWN","PO BOX ONE"
     : LPRINT "MARLOW,NH","ZIP Ø3456.ØØØ1
65262 LPRINT "TEL 6Ø3 446-3414"
     : IF X=7 END
65264 GOSUB 7
     : LPRINT
     : LPRINT "THANX NORLIN, MS."
     : USING
     : RETURN
65265 "L" X=7
     : GOTO "/
6527Ø "H" CLEAR
     : USING
     : A=14533
     : LPRINT TIME
     : Y=(PEEK (A+2))+3
     : LPRINT A;Y;Y
65271 "$" FOR I=A TO STATUS 2-3
     : IF PEEK I=13 AND PEEK (I-3)<>13 LET B=I
     : X=PEEK (I+3)+3
     : Y=Y+X
     : LPRINT B;X;Y
65272 NEXT I
     : LPRINT TIME
     : END
65277 " " USING
     : LPRINT TIME ," LOC.  LINE NO.
65278 FOR I=14533 TO STATUS 2
     : IF PEEK I=13 AND PEEK (I+4)=34 LPRINT
     CHR$ PEEK (I+5);I;PEEK (I+1);PEEK (I+2)
65279 NEXT I
     : LPRINT I;TIME
     : END
```

LINE	ANNOTATION
1	AUTO TIME
2	+2 BIO SBR
3	E 3 ERROR
4	O iF2 GLOBAL
5	P iF1 STATUS
6	# 6 FREE BYTES
7	T 7 ELAPSED TIME
8	W 8 AVE. TIME BANNERS
9	U 9 MONTHS
10	Y 10 COLORS
14	= PGM. NAME
20	M MENU

LINE	ANNOTATION
97	R. LINES 97-99 IDENTIFY REV. NO. OF PC-2
3000	% iiF1 PRINT A-Z
3100	3 iiF2 LOAD A-Z
3200	? 3200 PRINT ASCII NUMBER
4000	@ 4000 BANNER MISC.
4400	B BANNERS M/T
4500	Q 4500 BANNER TRS
4650	DATA
5000	V SHADOW PRINT
6005	J CALENDAR
6022	DATA
6400	6400 VALIDITY TRAP
8000	G LOAD L$(0-91)
8100	(Pi Sign) iiF3 PRINT L$(0-91)
9110	* iiiF1 9110 BIORHYTHM
61000	F PEEK
61400	Z POKE
62000	D SCRIP Y/N?
62010	J 62010 SCRIP
62080	X EXTRA COPIES
62500	C (EDIT) CHANGE SCRIP
64000	K USED IN SCRIP, BANNERS & SHADOW PRINT
64015	"RECIPE" 64015
64700	A ALARM
64900	N PRINT TIME
65200	iF4 time LCD
65220	S SIZES
65260	/ 65260 LABEL MFB
65265	L LABEL MFB
65270	H CHAR. PER LINE
65271	$ iiiF2 65271 C/COUNT A=(n . . . n) Y=PREV. CUMU
65277	SPACE DEFINED LABELS, GOTOs & GOSUBs

Math Teacher

Jack Corman
14 Eastlawn Drive
Hampton, VA 23664

When I purchased my TRS-80 pocket computer, I tried to find a way that it could serve everyone in the family. My sons were learning elementary math, so I wrote a program to sharpen their math skills.

The program will ask for a number between 1 and 10 to initialize the random number generator subroutine. Next, choose which math function you desire to practice: addition, subtraction, multiplication or division. Lastly, input the largest variable value that you can competently handle. There will be a 6-7 second pause as the computer runs its subroutine and formulates a problem for you to solve. To input your answer press ENTER, the digits of the answer and ENTER again. After each 7 correct responses you are asked if you desire to continue. A negative response will result in your percentage grade for the exercise.

```
2Ø   "A" CLEAR
     : PAUSE "***MATH TEACHER***"
3Ø   INPUT "ENTER NUMBER (1-1Ø) ";N
     : PAUSE "SELECT ONE OF"
33   PAUSE "THE FOLLOWING"
     : PAUSE "ADDITION (A)"
     : PAUSE "SUBTRATION (S)"
35   PAUSE "MULTIPLICATION (M)"
     : PAUSE "DIVISION (D)"
     : INPUT "A/S/M/D? ";L$
37   PAUSE ENTER LARGEST DESIRED"
     : INPUT "VARIABLE VALUE ";E
4Ø   IF L$="A" THEN 1ØØ
5Ø   IF L$="S" THEN 13Ø
6Ø   IF L$="M" THEN 16Ø
7Ø   GOSUB 5ØØ
8Ø   O=A*B
     : D=B
9Ø   H=H+1
     : PRINT O;" /  ";A;"=(?)"
     : GOTO 19Ø
1ØØ  GOSUB 5ØØ
11Ø  D=A+B
12Ø  H=H+1
     : PRINT A;" +  ";B;"=(?)"
     : GOTO 19Ø
13Ø  GOSUB 5ØØ
14Ø  O=A+B
     : D=B
15Ø  H=H+1
     : PRINT O;" -  ";A;"=(?)"
     : GOTO 19Ø
16Ø  GOSUB 5ØØ
17Ø  D=A*B
18Ø  H=H+1
     : PRINT A;" X  ";B;"=(?)"
19Ø  INPUT C
     : IF C<>D THEN 23Ø
2ØØ  BEEP 1
     : PAUSE "YOU GOT IT!"
     : I=I+1
     : T=T+1
21Ø  IF T=7 THEN 28Ø
22Ø  PAUSE "TRY THIS ONE..."
     : GOTO 4Ø
23Ø  PAUSE "***WRONG***"
     : PAUSE "THINK HARD"
     : PAUSE "TRY AGAIN"
24Ø  IF L$="A" THEN 12Ø
25Ø  IF L$="S" THEN 15Ø
26Ø  IF L$="M" THEN 18Ø
27Ø  GOTO 9Ø
28Ø  PAUSE "THIS IS FUN!"
     : INPUT "WANT TO CONTINUE? Y/N ";V$
29Ø  IF V$="Y"TO=O
     : GOTO 4Ø
3ØØ  U=I/H*1ØØ
     : USING "#####"
     : PRINT "YOUR GRADE IS";U;" %"
31Ø  END
5ØØ  A=O
     : B=O
51Ø  X=199Ø17
     : Y=1E4
     : N=((24298*N)+99991)/X
     : Z=N-INT N
52Ø  N=Z*X
     Z=INT(N/X*Y)/Y
     R=INT(Z*E+1)
53Ø  IF A=O THEN 55Ø
54Ø  B=R
     : RETURN
55Ø  A=R
     : GOTO 51Ø
```

Roots of General Equations

Ron Beahaars
615 Old Hackett Road
Greenwood, AR 72936

Here is a program capable of solving any general equation of the form y = f(x).

This program was introduced to me by Dr. Donald O. Pederson, Chairman of the Physics Department of the University of Arkansas at Fayetteville. I have adapted it in working form for use on the Pocket Computer (PC1). It should run on any TRS-80 system with little or no modification.

Many functions are of such complexity that mere algebra is not sufficient to yield numerical solutions to them. This program employs a numerical method which yields roots with a reasonable degree of accuracy.

The program's method is to make an educated guess at the root's value, a value of x which makes y equal to zero and as such is symbolized on a graph of the function as a crossing of the x-axis. The program starts with a value of x supplied by the user and increments it by a value, also provided by the user, until the sign of y changes. At this value of x it is evident that an axis crossing has occurred, between the current value of x and the previous value. The program uses this zero crossing as its next guess and continues until the root has been determined to within the desired degree of accuracy.

I will now briefly describe the workings of the program.

The operator is reminded at lines 2 and 3 to enter the function on lines 10, 30, and 120 in terms of the specified variables (i.e., at line 10, Y = F(X), at line 30, F = F(B), and at line 120, G = F(C)). Once this has been done, the user can return to RUN mode and type RUN "DONE" and ENTER and execution moves on. The program then asks the user to supply an initial value of X, an increment value, and an acceptable error at line 7. At line 10 the function is evaluated for the initial X to see if a root has been immediately found. If not, X is incremented and the function is re-evaluated for this new value (B) at line 30. If a root has not been found, at line 50 a check is made to see if the axis has been crossed (if so, Y and F would have different signs and would therefore have a negative product). Line 80 reverses the direction of search if I = 1 (this occurs only on the first interaction) and if F is farther from the axis than was Y.

If the axis has not been crossed, X is incremented repeatedly until crossing occurs, and when this happens there must be a root in the interval between the current value of X and the previous value, assuming the function is continuous in that region. Once axis-crossing is established, the location of the root must be found to within plus or minus E.

The method used assumes a linear variation between X and B and estimates the value of the root (C) based on this. It uses this value to generate new values of X and B so that the corresponding values of Y and F are on opposite sides of the axis. Line 120 is derived from the formula for a point where the line between X and B crosses the axis.

Line 170 assigns C as the new X value if G is on the same side of the axis as Y; line 180 assigns C as the new value of B if G and Y are on opposite sides of the axis.

The program is most efficient if the user has some knowledge of the graph of the function being evaluated, so that he can supply a starting value of the X and an increment which requires the shortest possible execution time. However, the program can be used to search for roots. For example, if the user knows that the root of some function is somewhere between -50 and zero, he may enter an initial X of -50 and an increment of 1 or two. The user should be advised that this application may be relatively time-consuming (execution times in excess of three minutes are possible in extreme cases).

General usage notes:

- A sufficient value for acceptable error is 1 × 10-6.

- If f(×) contains any trigonometric operators, it is imperative that the computer is placed in the RAD mode before execution begins. Otherwise, the program will yield incorrect answers.

- As the program runs, it will pause the incremented values of x as it determines the value of y. Once the x values are no longer briefly displayed, the program has found the general location of the root and will shortly provide the answer.

- The program uses roughly 530 steps of program memory once the function has been entered at lines 10, 30, and 120.

```
1    PAUSE "ROOTS OF EQUATIONS"
2    PAUSE "ENTER FUNCTION ON LINES"
3    PRINT "1Ø, 3Ø, AND 12Ø."
4    "DONE" PRINT "ENTER INIT. X, INCREMENT"
5    PRINT "AND ACCEPTABLE ERROR."
6    CLEAR
7    INPUT "X(I) : ";X, "INCREMENT : ";N, "ACC. ERROR
     : ";E
1Ø   Y=F(X)
     : I=1
     : Q=X
2Ø   IF Y=Ø GOTO 5ØØ
3Ø   B=X+N
     : F=F(B)
     : Q=B
4Ø   IF F=Ø GOTO 5ØØ
5Ø   IF Y*F<Ø GOTO 11Ø
6Ø   IF I<>1 GOTO 1ØØ
7Ø   IF ABS(F)<=ABS(Y) GOTO 9Ø
8Ø   N= -N
9Ø   I=2
1ØØ  X=B
     : PAUSE X
     : Y=F
     : GOTO 3Ø
11Ø  Q=X
     : IF ABS(X-B)<E GOTO 5ØØ
12Ø  C=X-Y*(B-X)/(F-Y)
     : G=F(C)
13Ø  Q=X
     : IF C=X GOTO 5ØØ
14Ø  Q=B
     : IF C=B GOTO 5ØØ
15Ø  Q=C
     : IF G=Ø GOTO 5ØØ
16Ø  IF Y*G<Ø GOTO 18Ø
17Ø  X=C
     : Y=G
     : GOTO 11Ø
18Ø  B=C
     : F=G
     : GOTO 11Ø
5ØØ  BEEP 1
     : PRINT "ROOT = ";Q
5Ø1  GOTO 6
51Ø  END
```

OS9 Assembly Language Programming

by Earl Bollinger

Developing computer programs in assembly language for use with the OS9 disk operating system requires different styles and techniques from that used normally with the TRS-80 Color Computer. The very nature of OS9 requires one to utilize more advanced programming concepts. Programs or subroutines for OS9 must be written in Position Independent Code. Also if the program or subroutine is expected to be used in a multiuser environment then it must also be coded so as to be reentrant.

Position Independent or Relocatable programs are simply programs, that, after they are assembled, can be loaded anyplace in memory and still be executable without reassembling them. The reason that programs must be position-independent is that OS9 dynamically allocates memory for the program at run time. OS9 assigns programs or modules memory on a first come, first served basis. Thus one never knows where OS9 will actually put your program when it is time to execute it. Necessarily, this does not mean that you'll be writing your programs differently; it's just that you'll have some constraints that have to be followed.

Writing programs to be position independent is relatively simple. One only has to avoid using any of the addressing modes that refer directly to a memory location. Instead of using a Jump to Subroutine (JSR), use Branch commands such as BSR or LBSR (Long Branch to Subroutine). Instead of JMP (unconditional Jump), use BRA or LBRA (Branch or Long Branch Always). Other more advanced techniques take advantage of the various indexed addressing modes, and the Program Counter Relative instructions that are available with the MC6809 microprocessor.

Programs that are designed to be reentrant are used in a multiuser environment. This only means that all your temporary data and variable storage areas must be located on the stack. Reason being, if you used the normal method of assigning data to specific memory locations, then, if more than one user attempts to use your program at the same time, the data and variable areas will quickly become corrupted. OS9 loads only one copy of a given program into memory at a time. It then lets all the users, who want to, share that program. The way it does this is by allocating to each user their own stack space in memory.

The OS9 Interactive Assembler is designed to aid you in developing programs for the OS9 system. The assembler uses a sort of free format source listing, outputs a formatted source listing and an object file from that original source program that it read. The actual source listing that the assembler reads is not formatted or pretty printed.

Consider the following source code segment:

```
*
* AN EXAMPLE SOURCE CODE PROGRAM SEGMENT
*
DATA1 RMB 2 LENGTH OF DATA LINE
DATA2 RMB 2 LENGTH OF FIELD ONE
DATA3 RMB 2 LENGTH OF FIELD TWO
DATA4 RMB 2 LENGTH OF FIELD THREE
STKEND RMB 200 HARDWARE DATA STACK AREA
STACK EQU .-1
ENTRY BSR INIT INITIALIZE PROGRAM
  LBSR GETDAT GET DATA
  LBSR PRODAT PROCESS DATA
  LBNE ERROR IF ERROR ABORT
LOOP1 LBSR GETDAT GET DATA
  LBSR PRODAT PROCESS DAT
  LBNE ERROR
  BCC OUIT IF DONE
  BRA LOOP1 CONTINUE UNTIL DONE
OUIT OS9 F$EXIT OUIT AND RETURN TO SYSTEM
```

The source program segment above is what the assembler expects to read. A character in the first line column of any line is considered a label, unless it is an asterisk; then it is a comment line. A space and then a character on any line is considered a machine language mnemonic operations code (opcode). After an opcode, you can enter another space and an operand for the opcode. Another space after the operand and you can have a comment, if desired.

Thus the fields that the assembler would normally use are:

(*) Comment line

LABEL (SPACE)OPCODE(SPACE)OPERAND-(SPACE)Optional comment section

(SPACE)OPCODE(SPACE)OPERAND(SPACE) Optional comment line

(SPACE)OPCODE(SPACE) Optional comment line if there is no operand for the OPCODE (eg: RTS, NOP)

The assembler expects you to separate all your various items with a space. The assembler will format the output source listing for you when you use it to assemble the program. Extra spaces and such will be included also in your output listing when the assembler processes your program. Actually, when you are using the OS9 Editor, the above source code lines are simply designed to make it easier to enter the program lines.

Assembling your programs can be a little confusing. Thus I'll present some OS9 Assembler commands with options and explanations of what it is doing.

OS9:ASM filename[ENTER]

This command will simply cause the assembler to process your program. It will not generate an output listing or an output object (binary) file. Its main use is in test assemblies of programs, when you are simply looking for minor typos or problem areas and such.

OS9:ASM src_program L[ENTER]

This command is essentially the same as the previous one, except that the 'L' option was invoked. This tells the assembler to generate a formatted output listing on the standard output path (your video screen normally).

OS9:ASM src_program L >/P >>/P[ENTER]

To redirect the formatted output listing to the printer, you can use the '>' output redirection modifier. If you want the errors also in the listing you must use the '>>' error redirection modifier; otherwise, the errors found will be displayed on your video screen instead of the printer.

OS9:ASM program L >/file >>/file[ENTER]

This command causes the assembly listing to be output to a file on disk. Also the errors, if any, are embedded into the listing. You can use a /pathlist/filename if desired.

OS9:ASM /d1/src_program[ENTER]

You can also tell the assembler to read its source input from a disk other than drive 0.

OS9:ASM src_program L O[ENTER]

This command tells the assembler to generate a formatted output listing on the standard output path (video screen), and to, also, create a binary object file (executable program) using 'src_program' as its filename and put the object file in the current execution directory (if you haven't changed it, the current execution directory will be /D0/CMDS). If any errors are incurred during assembly, the binary object file will automatically be rendered unuseable. Normally, the assembler will simply write in the object file a message stating that errors have occurred. Thus any attempts at executing a program that had errors will fail.

OS9:ASM /d3/prog L O=/d2/test #16k >/P >>/P[ENTER]

In this example, the assembler is to read its input from an assembly source program located on drive 3, output its formatted program listing to the printer, and put the binary object file on drive 2 under the name of "test". The #16k command option tells the system that 16k bytes of memory is to be allocated to the assembler for this task. The assembler will use much of the 16k bytes for its symbol table storage area. If you leave out the #Nk option, the system will allocate 4k bytes to the assembler anyway. The assembler can put about 270 symbolic references into the symbol table with 4k bytes allocated. For every 1k bytes of extra memory allocated, the assembler can add 93 additional symbol name entries to the symbol table. Every symbolic name entry in the assembler's symbol table requires 11 bytes of storage.

Normally, one uses the #Nk memory allocation option for large assembly programs and when using the OS9 assembly DEFinitionS files. These definitions files are normally located under the /D0/DEFS directory. The most common and generally used file being /D0/DEFS/OS9defs. If you list the OS9defs file, you notice that is contains a large set of predefined EQUate statements. These are provided in order to make the task of assembly language programming a little easier. But to take advantage of them will require using the #Nk option set to about #10k, as there are a lot of symbols in this file for the assembler to process.

OS9:ASM #10k test L O S[ENTER]

This example causes the assembler to read and assemble a source program named 'test'. The assembler is also told to generate a formatted output listing onto the standard output path (video screen). The 'O' option also causes a binary object file to be generated and put into the current execution directory (normally /D0/CMDS). The 'S' option tells the assembler to dump the symbol table onto the standard output path. The program memory allocation option #10k was in this case put on the command line right after 'ASM'.

The assembler will allow you to have a considerable amount of leeway in entering commands and options. One does not necessarily have to enter everything in a rigid fixed manner. One has to enter the 'ASM' in order to invoke the assembler first. Then you would enter the /pathlist/source_filename next, so that the assembler knows which file to assemble. Next, one can enter most any combination of options from the 'OPT' assembler commands. If you want, you can also redirect the outputs generated to a variety of places, such as another file, to the printer, or to the screen, etc..

Balloon Dart Toss

Paul Riches
1628 Le Gaye Dr.
Cardiff, CA 92007

Most games now days involve spaceships and lasers. So, I decided to make a unique game that is a little different.

This game involves throwing a dart at a randomly placed and colored balloon. You will use the left and right arrow keys to move, and the spacebar to fire. You must also get under the balloon before it explodes by itself. This involves both quickness and accuracy. If you are fast enough, you can shoot twice. To win the game you must do this 10 times in a row.

```
10 CLS
20 REM 7/12/82 BY PAUL RICHES
30 REM ***BALLOON***
40 PMODE 3, 1
     : PCLS
     : SCREEN 1, 0
70 X=128
80 FOR D=1 TO 10
90 PCLS
100 A=RND(255)
110 CIRCLE (A, 80), 25
120 LINE (A, 105)-(A, 130), PSET
130 B=RND(3) + 1
140 PAINT(A, 81), B, 4
150 FOR L=1 TO 35
160 B$="BM" + STR$(X) + ", I70; G10; D10; R10; U10;
    H10"
170 DRAW"C4;" + B$
180 A$=INKEY$
190 IF A$=CHR$(9) THEN GOSUB 430
     : X=X + 10
200 IF A$=CHR$(8) THEN GOSUB 430
     : X=X - 10
210 IF A$<>" " THEN 340
220 FOR N=191 TO 80 STEP -10
```

```
230 PLAY "T" + STR$(N) + ";" + STR$( INT((N - 70) /
    10))
240 CIRCLE (X, N), 9
250 NEXT N
260 IF ABS(X - A) < 6 THEN 270 ELSE 340
270 PCLS
280 PMODE 4, 1
    : SCREEN 1, 1
290 FOR M=1 TO 30
300 PLAY "T255;G"
310 CIRCLE (A, 80), M*2
320 NEXT M
    : PMODE 3, 1
    : SCREEN 1, 0
330 GOTO 360
340 NEXT L
350 GOTO 390
360 NEXT E
370 PRINT "YOU WON"
380 PLAY "T10; D; F; E; A; C; F; B; C"
390 REM END OF GAME
400 PRINT "YOU POPPED" D "BALLOONS"
410 PRINT "GAME OVER"
420 END
430 DRAW "C1;" + B$
440 RETURN
```

Animation

Andy Haff
Poway, California

Here is a relatively simple program that demonstrates the animation potential of Extended Color BASIC by revolving a spacepod around a planet. The program starts by "getting" the image of the spacepod and placing it in a 20×24 array. The background scenery is then set, prolonging the SCREEN 1, 1 command until after the user has input the revolution speed (of which there are three choices.) The program then proceeds by entering into the sin-cos loop which calculates the circular path for the X and Y coordinates of the pod.

Each time the new position of the pod is set, the last image is erased due to the overlapping of the new pod and the stored four pixel space surrounding it. The smooth motion is achieved by "putting" the stored image without having to draw it over and over again. Feel free to experiment with the program by changing the image of the pod or by adding sound.

```
10 ' REVOLVING SPACEPOD
20 ' ANDY HAFF
30 PMODE 4, 1
    : PCLEAR 4
40 A=128
    : B=92
    : R=70
    : PI=3.1415927
50 DIM S(20,24)
60 DRAW "BM8,4 G4 D8 R4 G4 R12 H4 R4 U8 H4 L4"
70 GET (0,0)-(20,24), S, G
80 PCLS
90 CIRCLE (A, B), 24, 5
    : PAINT (A, B), 5, 5
100 CIRCLE (A, B), 34, , .35
110 CIRCLE (A, B), 31, , .36
120 CIRCLE (A, B), 39, , .35
130 FOR X=1 TO 8
    : READ X1, Y1
140 CIRCLE (X1, Y1), 2
150 NEXT X
160 INPUT "INPUT REVOLUTION SPEED (1-3)"; SS
170 SCREEN 1, 1
180 FOR Z=1 TO 360 STEP SS
    : C=Z
190 C=C * PI / 180
200 X=INT( A - 4 + R * COS( C))
    : Y=INT( B - 4 + R * SIN( C))
210 PUT (X, Y)-(X+20, Y+24), S, PSET
220 NEXT Z
    : GOTO 180
230 DATA 232, 12, 76, 20, 20, 132, 8.34, 236, 100,
    24, 180, 240, 122, 180, 100
```

Extended BASIC Graphics Reverse

Scott Gunn
16 Westwood Drive
Fort Madison, Iowa 52627

This program places a machine language routine into memory starting at the address defined in line 10. The ML routine "instantly" reverses the colors on the first four pages of graphics memory. In four color modes it will change red to green and blue to yellow, and vice-versa. Before typing this program in, 16K users will have to type CLEAR 1, &H3EFF, while 32K users will have to type CLEAR 1, &H7EFF. Here is the program:

```
10 X=&H7F00
20 Y=X + 16
30 DATA 134, 255, 142, 6, 0, 160
40 DATA 132, 167, 128, 134, 255
50 DATA 140, 30, 0, 38, 245, 57, 2023
60 FOR A=X TO Y
70 READ D
80 T=T + D
90 POKE A, D
100 NEXT A
110 READ D
    : IF T<>D THEN PRINT "DATA ERROR -- CHECK DATA!"
120 END
```

16K users will have to change line 10 to read:

```
10 X=&H3F00
```

The following program demonstrates the first program's ML routine. This subroutine (or any other) can only be used after the routine has been POKEd into memory.

```
1000 PMODE 4, 1
    : PCLS
    : SCREEN 1, 1 'SETS UP SCREEN
1010 FOR X=0 TO 95 STEP 5 'PUTS CIRCLES ON SCREEN
1020 CIRCLE (128, 96), X
1030 NEXT X
1040 FOR X=1 TO 96 STEP 10 'MAKES BULLSEYE PATTERN
1050 PAINT (128, 96+X), 1, 1
1060 NEXT X
1070 EXEC &H7F00 '16K USERS CHANGE TO &H3F00
1080 A$=INKEY$
    : IF A$="" THEN 1080
1090 GOTO 1070
```

To use the ML routine, type (or use in a program line) "EXEC &H7F00" or "EXEC &H3F00", depending on whether you use a 32K or 16K computer respectively.

Directory to Printer

Alexander Benenson
585 West End Avenue
New York, NY 10024

Here is a short BASIC program that many Color Computer disk users will find useful—it does a directory to the printer. The directory will appear exactly as it does when the DIR statement is used. Below is the program for placing the routine into high RAM. Change the numbers in 10 and 20 to &H7FF7 and &H7FF8 if you have 32K.

```
10 CLEAR 200, &H3FF7
20 T=&H3FF8
30 FOR M=0 TO 7
40 READ X
50 POKE T + M, X
60 NEXT
70 DATA 134, 254, 151, 111, 189, 203, 207, 57
```

The following line will perform the actual printing from within a BASIC program:

80 EXEC T

Use EXEC T if you want a printed directory from the command level. Remember that other programs and statements could change or erase the value of T; in these cases, use EXEC &H3FF8 (or EXEC &H7FF8 if you have a 32K system.)

If you have a multi-drive system, it will be necessary to use the DRIVE statement first to get directories from drives other than zero.

SAVESND

Jerry Miller
6 Lavender Road
Rocky Point, NY 11778

The following program, which requires a 16K Extended BASIC Color Computer, isn't meant to serve any useful purpose in its present state, other than being instructional and possibly entertaining. Interested users may find it helpful as a template for something fancier. (Creativity and a knowledge of machine language are essential.)

I had the idea for this program shortly after I discovered that MOTOR ON: AUDIO ON causes toggling of bit 0 of $FF20 and bit 1 of $FF22 ($FF20 is more sensitive.) However, I was unsuccessful until I stumbled across the fact that going from A/D to D/A necessitates some sort of reinitialization. This is inelegantly accomplished here through SOUND 255, 1.

The machine language program has two entry points—one stores data for the time-delay between toggles; the other reads back the stored data and translates it back into sound. Because both routines execute for the same number of clock cycles within their respective loops, the frequencies produced are the same as those that were input.

The program simply enters a "DO WHILE" type of loop (exit by holding down any key until the playback ends and the program returns to BASIC) which turns on the recorder for a few seconds, transfers the sound from tape to memory, shuts the recorder off, and then produces a noisy, distorted version of the sound. The noise can be reduced somewhat by adjusting the recorder volume. Usually, however, the sounds played back are easily recognizable, despite the noise.

The storage is very expensive—12.5K for a few seconds of sound—there is no way to improve significantly on this, but, at least for a fun program, who cares?! The text-screen and all 8 pages of graphics memory are used for data storage, so you will see a screenful of "garbage" while the program is running. I eventually plan to recover the stored data and analyze it—I admit that my idea is far-fetched, but maybe I can even find patterns in certain sounds that will enable me to design an algorithm (or at least a much more memory-efficient program) to mimic them.

Type PCLEAR 8: RUN (ENTER)—otherwise the storage of data will abort the initial run with ?SN ERROR IN LINE ——. (Or see "PCLEAR" in the June 1982 issue.)

```
10 PCLEAR 8
20 CLEAR 10, 16300
30 OATA 86, 2, 10, 8E, 3F, FF, CE, 4, 0, A7
40 DATA A4, 8E, 0, 0, 30, 3, B6, FF, 20, A1
50 DATA A4, 26, 4, A7, A4, 20, F3, AF, C1, 10
60 DATA 21, 0, 0, 11, 83, 36, 0, 26, E2, 39
70 DATA CE, 4, 0, AE, C1, B6, FF, 20, 88, FC
80 DATA B7, FF, 20, 30, 1F, 26, FC, 11, 83, 36
90 DATA 0, 26, EC, 39
95 IF INKEY$<>"" THEN 95
100 FOR I = 16301 TO 16364
     : READ B$
     : POKE I, VAL( "$H"+B$ )
     : NEXT
110 MOTOR ON
     : AUDIO ON
     : EXEC 16301
     : MOTOR OFF
     : SOUND 255, 1
     : POKE 65315, 63
     : EXEC 16341
     : AUDIO OFF
120 IF INKEY$ = "" THEN 110
```

ASSEMBLY LANGUAGE PROGRAM

```
     LDA   #2
     LDY   #16383
     LDU   #1024
B1   STA   ,Y
     LDX   #0
B2   LEAX  3,X
     LDA   65312
     CMPA  ,Y
     BNE   B3
     STA   ,Y
     BRA   B2
B3   STX   ,U++
     LBRN  B4
B4   CMPU  #13824
     BNE   B1
     RTS
     LDU   #1024
B5   LDX   ,U++
     LOA   65312
     EORA  #252
     STA   65312
B6   LEAX  -1,X
     BNE   B6
     CMPU  #13824
     BNE   B5
     RTS
```

Reverse Characters

Howard Drake
P.O. Box 375
Canyon, TX 79015

The following short program shows how to reverse normal Color Computer screen printing by PEEKing into the video and replacing upper case with "lower case" letters. The effect is almost as interesting as the use of the PEEK function in games and possibly printing from the screen.

POKEing memory location 65314 with the number 8 changes the screen to orange and black rather than green and black. POKEing a 7 into the same location changes the screen back to green and black.

Lines 1045 and 1060 in the program are used as timers to slow the flash rate.

Just type in the program, press the CLEAR key, LIST the program and RUN it.

```
999 REM REVERSE CHARACTERS
1000 FOR ZZ=1024 TO 1535
1010 Z=PEEK(ZZ)
1020 IF Z>64 THEN POKE ZZ, Z-64
1030 NEXT ZZ
1035 REM FLASH ORANGE AND GREEN
1040 POKE 65314, 7
1045 SOUND RND(200), 1
1050 POKE 65314, 8
1060 SOUND RND(100), 1
1070 GOTO 1040
```

Hints and Tips

Charles B Levinski
10 Southside Avenue
South River, NJ 08882

Below are several hints and tips which I have found useful for the Color Computer. I should point out that some of these hints are useable only with Extended Color BASIC.

On page 20 of the February, 1982 Microcomputer News there is an article on TAB and PRINTUSING. Most of these routines make use of the following statement:

```
PRINT TAB(N) USING A$; A
```

Unfortunately, this yields a syntax error when used on the Color Computer. What will work, however, is:

```
PRINT TAB(N);: PRINT USING A$; A
```

While not as simple as the former statement, it is equally effective. To use this statement with a printer, replace PRINT with PRINT #-2,.

I was quite enthused with the suggestion in the same article to incorporate line feeds in the USING string. The recommended technique was to press the line feed (down arrow) key at the appropriate point in the string.

Uh-Oh! For some reason, the down arrow will not store as part of the USING string, even though the down arrow does generate a decimal 10, or line feed. In order to overcome this and still accomplish page formatting, try the following:

```
10 CLS
20 B$="SEE?"
30 A$="IT WORKS QUITE" + CHR$(13) + "WELL-%  %"
40 PRINT USING A$; B$
```

The CHR$(13) in line 30 causes a carriage return (and line feed) to be generated, performing exactly the function described in the February News.

Lastly, I use a Line Printer VII, which has only a manual paper advance. I don't really care for manually advancing the paper, and after printing a report, I frequently wish to advance the paper to a point where it can be torn at the perforations.

The following routine at the end of the printing routine accomplishes this nicely:

```
9000 CLS
     : PRINT "PRESS <A> TO ADVANCE PAPER"
     : PRINT "PRESS <I> TO CONTINUE"
9010 A=PEEK(339)
     : IF A>254 OR A<253 THEN 9010
9020 IF A=254 THEN PRINT#-2,
     : GOTO 9010
```

The printer will continue to advance the paper until the A key is released.

Pressing I will exit the subroutine and allow execution of the next program line.

I hope these tips are useful to some other Color Computer owners.

Line Printer Width

Tom Garcie
Box 15186
Tucson, AZ 85708

Thanks for the "Line Printer Width Driver" program for the Color Computer as published in the April, 1982 issue of *TRS-80 Microcomputer News*. It fills a need for me in that I write programs for publication in various magazines and want the program listings to be just 32 characters per line, as in a screen listing.

I just can't help trying to "improve" on other people's programs . . . So, I made a few changes . . . Mostly, I put the program up near the top of my (32K) memory where I would always want it. I also have the program automatically delete itself after running.

```
1 DATA 182, 001, 103, 167, 141, 000, 046, 190, 001,
     104, 175, 141, 000, 040, 134, 126, 183, 001,
     103, 048, 141, 000, 004, 191, 001, 104, 057, 052
2 DATA 002, 150, 111, 129, 254, 038, 016, 150, 156,
     139, 001, 145, 155, 037, 008, 015, 156, 134,
     013, 173, 159, 160, 002, 053, 002, 018, 018, 018
3 ST=32710
4 FOR AD=ST TO ST + 55
5 READ CO
6 POKE AD, CO
7 NEXT AD
8 EXEC 32710
9 CLEAR 200, 32709
10 POKE 155, 33
11 DEL 1 - 11
```

St. Patrick's Day

Tom Flook and Ken Jones
11870 Heidelberg Ln., Lot 33
Whitmore Lake, MI 48189

Here is a special treat for the Irish and for all those who "feel" Irish on St. Patrick's Day.

```
1Ø '  ST. PATRICKS DAY DEMO
2Ø '
3Ø 'BY: TOM FLOOX AND KEN JONES
4Ø 'JAN 17, 1982
5Ø '
6Ø 'FOR 16X EXTENDED BASIC COLOR COMPUTER
7Ø CLS
     : PRINT @ 169, "-PRESS ANY KEY-"
     : PRINT @ 229, "FOR A HOLIDAY MESSAGE"
75 PRINT @ 295, "FROM--FLOOK AND JONES"
8Ø A$=INKEY$
     : IF A$="" GOTO 8Ø
9Ø CLEAR 8ØØ
1ØØ PMODE 3, 1
     : PCLS3
     : SCREEN 1, Ø
11Ø GOSUB 13Ø
     : GOSUB 25Ø
     : GOSUB 55Ø
     : GOSUB 1ØØØ
     : GOSUB 6ØØ
     : GOSUB 33Ø
     : GOSUB 71Ø
     : GOSUB 1Ø7Ø
     : GOSUB 65Ø
12Ø GOTO 1Ø
13Ø REM*******CLOVER*******
14Ø Y=148
     : FOR X=4Ø TO 6Ø STEP 2Ø
     : CIRCLE (X, Y), 15, 1
     : PAINT (X, Y), 1
     : NEXT
15Ø X=86
     : FOR Y=11Ø TO 13Ø STEP 2Ø
     : CIRCLE (X, Y), 15, 1
     : PAINT (X, Y), 1
     : NEXT
16Ø Y=9Ø
     : FOR X=4Ø TO 6Ø STEP 2Ø
     : CIRCLE (X, Y), 15, 1
     : PAINT (X, Y), 1
     : NEXT
17Ø X=14
     : FOR Y=11Ø TO 13Ø STEP 2Ø
     : CIRCLE (X, Y), 15, 1
     : PAINT (X, Y), 1
     : NEXT
18Ø CIRCLE (5Ø, 12Ø), 3Ø, 1
     : PAINT (5Ø, 12Ø), 1
19Ø FOR Y=1 TO 5
2ØØ CIRCLE (3Ø + Y, 146), 45, 1, 1, .75, .25
21Ø NEXT Y
22Ø MH$="L16BR4U1ØR8D1ØBL2U4L4D4"
23Ø DRAW "BM198, 92; C2; S8;" + MH$
     : PAINT (176, 9Ø), 1, 2
24Ø RETURN
25Ø REM*******MESSAGE*******
26Ø DRAW "C2; S8; BM9Ø, 2Ø"
27Ø AA$="HAPPY"
28Ø GOSUB 12ØØ
29Ø DRAW "C2; S8; BM15, 5Ø"
3ØØ AA$="ST. PATRICXS DAY"
31Ø GOSUB 12ØØ
32Ø RETURN
33Ø REM*******MAN-OOWN*******
34Ø MF$="D6R3U6BR5D6R3U6BD6R1D1R2D1R2D1L9U3R1BL
     5R1D3L9U1R2U1R2U1"
35Ø MP$="G3R1D1R1D1R1D1R4U1R1U1R2D1R1D1R4U1R1U1R1U1R
     1H3"
36Ø MB$="L3U1R3U1R4D1R3D1L3D1L4"
37Ø MS$="U4F3G3BD2E4H4L3U2G3H3D2L3G4F4U2H3E3D4"
38Ø M$="D3F3R3E3U3BD2BL2H1BL2G1BD2BR1R2"
39Ø S1$="BD6BL12"
4ØØ S2$="BL4"
41Ø S3$="BD2BR4"
42Ø S4$="BD9BR4"
43Ø DRAW "BM17Ø, 94; C2; S8;" + MF$
44Ø DRAW "BM17Ø, 94; C3; S8;" + MF$
45Ø DRAW "BM169, 94; C2; S8;" + MP$ + S1$ + MF$
46Ø DRAW "BM169, 94; C3; S8;" + MP$ + S1$ + MF$
47Ø DRAW "BM176, 9B; C2; S8;" + MB$ + S2$ + MP$ + S1$
     + MF$
48Ø DRAW "BM176, 9B; C3; S8;" + MB$ + S2$ + MP$ + S1$
     + MF$
49Ø DRAW "BM193, 1Ø5; C2; S8;" + MS$ + S3$ + MB$ +
     S2$ + MP$ + S1$ + MF$
5ØØ DRAW "BM193, 1Ø5; C3; S8;" + MS$ + S3$ + MB$ +
     S2$ + MP$ + S1$ + MF$
51Ø DRAW "BM172, 94; C2; S8;" + M$ + S4$ + MS$ + S3$
     + MB$ + S2$ + MP$ + S1$ + MF$
52Ø PAINT (176, 13Ø), Ø, 2
     : PAINT (181, 88), Ø, 2
     : PAINT (181, 12Ø), 1, 2
     : PAINT (1B1, 115), Ø, 2
53Ø FOR DLY=1 TO 1ØØØ
     : NEXT
54Ø RETURN
55Ø REM*******EYES*******
56Ø Y=72
     : FOR X=5Ø TO 64 STEP 12
     : CIRCLE (X, Y), 9, 2
     : PAINT (X, Y), 2
     : CIRCLE (X, Y), 2, Ø
     : NEXT
57Ø LINE (52, 6Ø)-(46, 56), PSET
58Ø LINE (62, 6Ø)-(74, 56), PSET
     : X=5Ø
59Ø RETURN
6ØØ REM*******EYES MOVE OVER*******
61Ø CIRCLE (X, Y), 2, 2
     : FOR Z=1 TO 5
62Ø CIRCLE (X + Z, Y), 2, Ø
     : CIRCLE (X + Z + 12, Y), 2, Ø
63Ø NEXT Z
64Ø RETURN
65Ø REM*******EYES MOVE BACX*******
66Ø FOR ZZ=5 TO 1 STEP -1
```

```
67Ø CIRCLE (55-ZZ, Y), 2, 2
     : CIRCLE (69-ZZ, Y), 2, 2
     : NEXT ZZ
68Ø FOR ZZ=1 TO 5
69Ø CIRCLE (5Ø, 72-ZZ), 2, Ø
     : CIRCLE (62, 72-ZZ), 2, Ø
7ØØ NEXT ZZ
     : FOR D=1 TO 1ØØØ
     : NEXT D
     : RETURN
71Ø REM*******DANCE*******
72Ø FOR A=1 TO 2
73Ø RF$="D6R3U6BD6R1D1R2D1R2D1L9U3R1"
74Ø LF$="D6R3U6BD6R1D3L9U1R2U1R2U1R2U1"
75Ø RD$="G4H1G1H1G1F4E6U2"
76Ø LD$="F4E1F1E1F1G4H6U2"
77Ø DRAW "BM184, 138; C3; S8" + RF$
     : PLAY "V2ØT5D5L8C;"
78Ø DRAW "BM186, 138; C2; S8;" + RD$
     : PLAY "O4AFF"
79Ø DRAW "BM186, 138; C3; S8;" + RD$
     : PLAY "CFF;"
8ØØ DRAW "BM184, 138; C2; S8;" + RF$
     : PLAY "AFA"
81Ø DRAW "BM168, 138; C3; S8;" + LF$
     : PLAY "O5C; O4BA"
82Ø DRAW "BM17Ø, 14Ø; C2; S8;" + LD$
     : PLAY "BGG"
83Ø DRAW "BM17Ø, 14Ø; C3; S8;" + LD$
     : PLAY "EGG"
84Ø DRAW "BM168, 138; C2; S8;" + LF$
     : PLAY "BGB"
85Ø DRAW "BM184, 138; C3; S8;" + RF$
     : PLAY "O5DC; O4B;"
86Ø DRAW "BM186, 138; C2; S8;" + RD$
     : PLAY "AFF;"
87Ø DRAW "BM186, 138; C3; S8;" + RD$
     : PLAY "CFF;"
88Ø DRAW "BM184, 138; C2; S8;" + RF$
     : PLAY "AFA;"
89Ø DRAW "BM168, 138; C3; S8;" + LF$
     : PLAY "O5C; O4BA;"
9ØØ DRAW "BM17Ø, 14Ø; C2; S8;" + LD$
     : PLAY "BAB;"
91Ø DRAW "BM17Ø, 14Ø; C3; S8;" + LD$
     : PLAY "G; O5C; O4B;"
92Ø DRAW "BM168, 138; C2; S8;" + LF$
     : PLAY "AFF; L4FP1;"
93Ø NEXT A
94Ø RETURN
95Ø REM*******IR1SH JIG*******
96Ø JI$="V2ØT3O5L8C; O4AFF; CPF; AFA; O5CD4BA; BGG;
     EGG ;BGB; O5DC; O4B; AFF; CFF; AFA; O5C; O4BA;
     BAB; G; O5C ;O4B; AFP; L4F; P1;"
97Ø PLAY JI$
98Ø GOTO  96Ø
99Ø RETURN
1ØØØ REM******* I'M LOOKING OVER*******
1Ø1Ø LO$="V2ØT4; O4L2CL8GEDL2CL8DL2EGO5L8CO4L2BO5L8
     CL2DO4L8DEL2F#AO5DD"
     : LA$="O4L2BL8AGL2BL8AGL2AL8AAL2AO5L2DO4L8
     BAO5L2DO4L88AL2GL8GGG"
1Ø2Ø LB$= "O5L2DL8CO4L8BO5L8DL2CO4L8AL2BBL8BL2AL8GL2A
     L8BO5CL2DO4L2CL2D5CL8CP4"
1Ø3Ø X$=LO$ + LA$
1Ø4Ø X1$=LO$ + LB$
1Ø5Ø PLAY X$
     : PLAY X1$
1Ø6Ø RETURN
1Ø7Ø REM*******INTO HAT*******
1Ø8Ø PAINT (176, 13Ø), 3, 2
     : PAINT (181, 88), 3, 2
     : PAINT (181, 12Ø), 3, 2
     : PAINT (181, 115), 3, 2
1Ø9Ø DRAW "BM172, 94; C3; S8;" + M$ + S4$ + MS$ + S3$
     + MB$ + S2$ + MP$ + S1$ + MF$
11ØØ GIRGLE (173, 15Ø), 5, 3
111Ø DRAW "BM193, 1Ø5; C2; S8" + MS$ + S3$ + MB$ +
     S2$ + MF$ + S1$ + MF$
112Ø DRAW "BM193, 1Ø5; C3; S8;" + MS$ + S3$ + MB$ +
     S2$ + MF$ + S1$ + MF$
113Ø DRAW "BM176, 98; C2; S8;" + MB$ + S2$ + MP$ +
     S1$ + MF$
114Ø DRAW "BM176, 98; C3; S8;" + MB$ + S2$ + MP$ +
     S1$ + MF$
115Ø DRAW "BM169, 94; C2; S8;" + MF$ + S1$ + MF$
116Ø DRAW "BM169, 94; C3; S8;" + MF$ + S1$ + MF$
117Ø DRAW "BM17Ø, 94; C2; S8;" + MF$
118Ø DRAW "BM17Ø, 94; C3; S8;" + MF$
119Ø RETURN
12ØØ REM********LETTERING*******
121Ø FOR XX=1 TO  LEN(AA$)
122Ø RESTORE
     : LL=Ø
123Ø READ LL$, CC$
124Ø IF LL$=MID$(AA$, XX, 1) THEN DRAW CC$
     : GOTO 126Ø
125Ø LL=LL + 1
     : IF LL<13 THEN 123Ø
126Ø NEXT
     : RETURN
127Ø DATA " ", "BM + 7, Ø"
128Ø DATA "A", "U4; E2; F2; D2; NL4; D2; BM + 3, Ø"
129Ø DATA "C", "BM + 1, -Ø; H1; U4; E1; R2; F1; BM +
     Ø, + 4; C1; L2; BM + 6, Ø"
13ØØ DATA "D", "U6; R3; F1; D4; C1; L3; BM + 7, Ø"
131Ø DATA "H", "U3; NU3; R4; NU3; D3; BM + 3, Ø"
132Ø DATA "I", "BM + 1, Ø; R1; NR1; U6; NL1; R1 ;BM +
     4, + 6"
133Ø DATA "K", "U3; NU3; R1; NE3; F3; BM + 3, Ø"
134Ø DATA "P", "U6; R3; F1; D1; G1; L3; BM + 7, 3"
135Ø DATA "R", "U6; R3; F1; D1; C1; L2; NL1; F3 ;BM +
     3, Ø"
136Ø DATA "S", "BM + Ø, -1; F1; R2; E1; U1; H1; L2
     ;H1; U1; E1; R2 ; F1; BM + 3, + 5"
137Ø DATA "T", "BM + 2, + Ø; U6; NL2; R2: BM + 3, +
     6"
138Ø DATA "Y", "BM + Ø, -6; D2; F2; ND2; E2; U2 ;BM +
     3, 6"
139Ø DATA ".", "BM + 2, Ø; U1; BM + 5, + 1"
```

429 Computer Centers Nationwide

ALABAMA
BIRMINGHAM 2425 Green Springs Hwy 9233 Parkway East
HUNTSVILLE 1400 N Memorial Pkwy
MOBILE 405 Bel-Air Blvd
MONTGOMERY #24 Union Square S/C

ARIZONA
PHOENIX 10233 Metro Pkwy E 4301 N 7th Ave
SCOTTSDALE 2525 N Scottsdale Rd
TEMPE 83 E Broadway
TUCSON 5622 E Broadway Campbell Plaza 2830 N Campbell Ave

ARKAKSAS
LITTLE ROCK Town & Country S/C University & Asher

CALIFORNIA
ANAHEIM 509 Katella
BAKERSFIELD 2018 Chester Ave
BERKELEY 1922 Grove St
BEVERLY HILLS 8500 Wilshire Blvd
BREA Imperial Shopping Center, 391 South State College
CANOGA PARK 8375 Topanga Canyon
CARMICHAEL 6305 Fairoaks Blvd
CHICO 1834 Mangrove Ave
CHULA VISTA 1201 3rd Ave
CITRUS HEIGHTS 7405 Green Back Ln at San Juan
DOWNEY 8031 Florence Ave
EL TORO El Toro Rd at San Diego Freeway
ESCONDIDO 347 W Mission Ave
FREMONT Fremont Hub 39114 Fremont
FRESNO Princeton S/C, 2721 N Blackstone Ave
GARDEN GROVE 12821 Knott St
GLENDALE 236 N Brand Blvd
HAYWARD 24784 Hesperian Blvd
HOLLYWOOD 6922 Hollywood Blvd
IRVINE Rechtl at Main St
LAKEWOOD 5830 Lakewood Blvd
LA MESA 5348 Jackson Dr
LONG BEACH 2119 Bellflower Blvd
LOS ANGELES 740 S Olive St 5240 Century Blvd (Airport area)
MODESTO 221 McHenry Ave
MONTCLAIR 5237 Arrow Hwy
MONTEREY 484 Washington St
MOUNTAIN VIEW 1933 El Camino Real W
OAKLAND 1733 Broadway
PASADENA 575 S Lake Ave
PLEASANT HILL 508 G Contra Costa Blvd
RIVERSIDE 3844 La Sierra Ave
SACRAMENTO 4749 J St
SAN BERNARDINO 764 Inland Center Dr
SAN DIEGO 3082 Convement Dr , 3902 El Cajon Blvd
SAN FRANCISCO One Market Place, 2920 Geary Blvd
SAN JOSE 1228 S Bascom Ave
SAN MATEO 3180 Campus Dr
SANTA ANA 2320 S Fairview St (at Warner)
SANTA BARBARA 4141 State St A-1
SANTA FE SPRINGS 14138 East Firestone Blvd
SANTA MONICA 511 Wilshire Blvd
SANTA ROSA 823 4th St
SHERMAN OAKS 14936 Ventura Blvd
STOCKTON College Sq S/C 963 West Match Lane
TARZANA 18545 Ventura Blvd
TORRANCE 3840 Sepulveda at Hawthorne
VENTURA 4005 E Main St
WEST COVINA 2516 E Workman St

COLORADO
AURORA Hoffman Heights Shop Ctr
BOULDER Arapahoe Plaza, 3550 Arapahoe
COLORADO SPRINGS 4341 N Academy
DENVER 8080 E Quincy Green Briar Plaza 7075 Pecos
LAKEWOOD 2099 Wadsworth Blvd

CONNECTICUT
EAST HAVEN 51 Frontage Rd
FAIRFIELD 1196 Kings Hwy & Rt 1
HARTFORD The Richardson Bldg 942 Main St
MANCHESTER 228 Spencer St
NORWALK Rt 7 345 Main Ave
ORANGE Edwards Food Warehouse Shop Ctr 538 Boston Post Rd.
WATERBURY 105 Bank St
WATERFORD 122 Boston Post Rd
WEST HARTFORD 39 S. Main St

DELAWARE
DOVER Eggehill Shop Ctr Rt #13
WILMINGTON 3847 Kirkwood Hwy

DISTRICT OF COLUMBIA
N W. WASHINGTON 1800 M St 15th & L St , Van Ness Station, 4250 Connecticut Ave

FLORIDA
ALTAMONTE SPRINGS 756 B East Altamonte Dr
BOCA RATON 1662 N. Federal Hwy
CLEARWATER 2460-D US 19 North
DAYTONA BEACH Volusia Plaza
FT. LAUDERDALE 4368 N Federal Hwy
FORT MYERS Edison Mall
FORT PIERCE Center West Shopping Center, U S Hwy 1
GAINESVILLE 3315 Archer Rd
HOLLYWOOD 429 S. State Rd #7
JACKSONVILLE 8252 Arlington Expnwy , Roosevelt Mall, Roosevelt Blvd
LAKELAND 2810 S Florida Ave
LAUDERDALE LAKES 4317-25 N State Rd 7
MERRITT ISLAND 700 E Merritt Island Causeway
MIAMI 9499 S Dixie Hwy , 1601 Biscayne Blvd 15 SE 2nd Ave , 20761 S Dixie Hwy
N. MIAMI BEACH The Promenade, 1777 N E 163rd St
ORLANDO 1238 E Colonial Dr
PENSACOLA Eastgate Shopping Center, 6925 N 9th Ave
ST PETERSBURG 3451 66th St N
SARASOTA 5251 S Tamiami Tr (Hwy 41)
TALLAHASSEE 2529 S Adams
TAMPA 4555 W Kennedy, 1825 E Fowler Ave
W PALM BEACH 2271-A Palm Beach Lakes Blvd

GEORGIA
ATLANTA 2108 Henderson Mill 49 W Paces Ferry Axers Mill S/C 2937 Cobb Parkway NW , 113 Peachtree St
AUGUSTA 3435 Wrightsboro Rd
COLLEGE PARK 5309 Old National Hwy
COLUMBUS Columbus Square Mall, 3050 Macon Rd
DORAVILLE 5897 Buford Hwy
MACON 1467 Eisenhower Pkwy
SAVANNAH Chatham Plaza, 7805 Abercorn St

HAWAII
HONOLULU Pacific Trade Center-Ground Floor, 190 South King St

IDAHO
BOISE 891 S Capitol Blvd

ILLINOIS
AURORA 890 North Lake St
BLOOMINGTON 401 N Veterans Parkway (Eastland Commons)
CHICAGO 4355 S Archer Ave Chk Plaza 309 S Wabash 72 West Adams, 524 N Michigan Ave , 101 W Washington St at Clark
DEERFIELD Deerbrook Mall
ELMWOOD PARK 7212 W Grand Ave.
FAIRVIEW HEIGHTS #4 Market Place
HOMEWOOD/GLENWOOD 379 Glenwood Landing; 18230 E Halsted at 183rd
JOLIET 2415 W Jefferson Jl
LaGRANGE One S LaGrange Rd.
LIBERTYVILLE 1350 S Milwaukee Ave.
LOMBARD 4 Yorktown Center
MOLINE 4401 44th Ave
NILES 8349 Golf Rd
OAK LAWN 4815 W 95th St
PEORIA 4125 N Sheridan Rd
ROCKFORD North Town S/C, 3600 N Main St
SCHAUMBURG 651 Mall Dr
SPRINGFIELD Sherwood Plaza 2482 Wabash

INDIANA
EVANSVILLE 431 Diamond Ave
FT. WAYNE 247 Northcrest S/C
GRIFFITH 208 W Ridge Rd
HOBART Save-More Plaza Rtes. #6 & 51
INDIANAPOLIS 6242 E 82nd St Castleton Plz Speedway Plaza 6129 B Crawfordsville, 10013 E Washington St , Two W Washington St
SOUTH BEND 1827 South Bend Ave
TERRE HAUTE 3460 U S Hwy 41 S

IOWA
CEDAR RAPIDS 111 First Ave , S E (Downtown)
DAVENPORT 616 E Kimberly Rd
DES MOINES 7660 Hickman Rd Sherwood Forest S/M

KANSAS
OVERLAND PARK 8619 W 95th
TOPEKA White Lakes Plaza West Towel, 3715 Plaza Dr
WICHITA 2732 Blvd Plaza S/C

KENTUCKY
FLORENCE 7727 Mall Rd
LEXINGTON 2909 Richmond Rd
LOUISVILLE Louisville Galleria, 4133 Shelbyville Rd

LOUISIANA
ALEXANDRIA 1213 Texas Ave
BATON ROUGE 7007 Florida Blvd
GRETNA 400 La Palco Blvd
HOUMA 2348 W Park Ave (Hwy 24)
LAFAYETTE University Square at Congress Blvd
METAIRIE 3750 Veterans Hwy
NEW ORLEANS 322 St Charles Ave
SHREVEPORT 1545 Line Ave

MAINE
BANGOR Maine Square

MARYLAND
BALTIMORE 7942 Belair Rd Putty Hill Plaza, 115 N Charles St at Lexington
BETHESDA 7900 Wisconsin Ave
CATONSVILLE One Mile West S/C, 6600 B Balt Nat'l Pike
FREDERICK Shoppers World, Rt 40W
NEW CARROLLTON-LANHAM 7949 Annapolis Rd
PASADENA 8120 Ritchie Hwy
ROCKVILLE Congressional Plaza, 1673 Rockville Pike
SALISBURY Shoppers World S/C, Rt 50
TEMPLE HILLS 4520 St Barnabas Rd
TOWSON-LUTHERVILLE Yorktown S/C York Rd at Ridgley Rd

MASSACHUSETTS
BOSTON 730 Commonwealth Ave 111 Summer St 372 Boylston
BOSTON 372 Boylston
BRAINTREE South Shore Plaza 250 Granite St
BROCKTON 675 Belmont
BURLINGTON Crossroads Plaza Rt 3 S
CAMBRIDGE Harvard Square 28 Boylston St
CHESTNUT HILL 200 Boylston St
FALL RIVER 80 William S Canning Blvd
NATICK 1400 Worcestar Rd
SAUGUS 343 Broadway
SPRINGFIELD 1965 Main St Northgate Plz
WORCESTER Lincoln Plaza

MICHIGAN
ANN ARBOR 2515 Jackson Rd
BIRMINGHAM 3620 W Maple Rd
DEARBORN Westborn Shop Ctr 23161 Michigan Ave
DETROIT DWNTN 1559 Woodward Ave
FLINT 03298 Miller Rd , Yorkshire Plaza
GRAND RAPIDS 3142 28th St SE
KALAMAZOO 25 Kalamazoo Center
LANSING 2519 S Cedar St
LIVONIA 33470 W 7 Mile Rd
PLAINFIELD North Kent Mall
PONTIAC North Oak Plaza-Pontiac Mall 2436 Elizabeth Lake Rd
ROSEVILLE 31873 Gratiot Ave
SOUTHFIELD 17651 West 12 Mile Rd
TROY Oakland Plaza 322 John R Rd
WARREN 29038 Van Dyke Ave

MINNESOTA
BLOOMINGTON 10566 France Ave S
BRADLEY 7974-76 University Ave North
GOLDEN VALLEY Golden Valley S/C 8016 Olson Memorial Hwy
MINNEAPOLIS 830 Marquette Ave
ST. PAUL 6th & Wabasha

MISSISSIPPI
GULFPORT 516A Courthouse Rd
JACKSON 979 Ellis Ave

MISSOURI
DES-PERES 11960 Manchester Rd
FLORISSANT 47 Florissant Oaks S/C
INDEPENDENCE 1325 S Noland Rd
KANSAS CITY 4025 N Oak Trafficway
ST. ANN 10472 St Charles Rock Rd
ST LOUIS 500 No Broadway (Commerce Bank Bldg Downtown)
SPRINGFIELD 2684 S Glenstone

NEBRASKA
LINCOLN 4601 "O" St
OMAHA 3006 Dodge St , 1318 72nd St at Pacific

NEVADA
LAS VEGAS Commercial Center, 953 E Sahara #31-B
RENO 3326 Kietzke Lane

NEW HAMPSHIRE
MANCHESTER Hampshire Plaza, 1800 Elm St.
NASHUA 429 Amherst St., Rt. 101A

NEW JERSEY
BRIDGEWATER 1472 U.S. Highway 22 East
E. BRUNSWICK 595 A Rt 18
E HANOVER Rt 10 Hanover Plaza
LAWRENCEVILLE Rt 1 & Texas Ave
NEWARK 595 Broad
NORTHFIELD 322-24 Tilton Rd
PARAMUS 175 Rt 17 S
SPRINGFIELD Rt #22 Center IsN
TOMS RIVER 700 Rt 37 West
VOORHEES 35 Eagle Plaza

NEW MEXICO
ALBUQUERQUE 2108 San Mateo NE

NEW YORK
ALBANY Shoppers Plz Wolf Rd
BAYSHORE 1751 Sunrise Hwy
BETHPAGE 422 N Wantagh Ave
BROOKLYN 531 86th St
BUFFALO 839 Niagara Falls Blvd
FRESH MEADOWS 187 12 Horace Harding Exp
GARDEN CITY 960 Franklin Ave
JOHNSON CITY Giant Shopping Center Harry L Drive
KINGSTON Kings Mall, Rt 9W
MANHASSET 1550 Northern Blvd
MELVILLE 1SS Mall, Rt 110
NEWBURGH Zayre Plaza Rt #17K
NEW ROCHELLE 211 North Ave
NEW YORK 385 Fifth Ave 139 E 42nd St 19 W 23rd St 347 Madison Ave 270 Park Ave South, 1282 Broadway 9 Broadway
NIAGARA FALLS Pine Plaza 8351 Niagara Falls Blvd
NIEGO PARK 97-77 Queens Blvd
ROCHESTER 3000 Winton Rd
SCARSDALE 365 Central Park Ave
SCHENECTADY Woodlawn Plaza
SPRING VALLEY White House Center 88 W Rt 59
STATEN ISLAND 2408 Richmond Ave
SYRACUSE 2544 Erie Blvd Hotel Sylacuse 510 S Warren St
UTICA Riverside Mall
VALLEY STREAM Green Acres Shop Ctr
YONKERS Cross Country Shop Ctr

NORTH CAROLINA
ASHEVILLE K-Mart Shopping Center Tunnel Rd
CHARLOTTE 3732 Independence Blvd Tyvola Mall, 5401 South Blvd
DURHAM South Square Mall
FAYETTEVILLE Eutaw Shopping Center 815 Elm St Fayetteville Street Mall
GREENSBORO 3718 High Point Rd
RALEIGH Townridge Sq Hwy 70 W
WILMINGTON Independence Mall
WINSTON-SALEM 629 Peters Creek Pkwy

OHIO
AKRON Fairlawn Plaza, 2727 W Market St
BEDFORD HEIGHTS 5217 Northfield Rd
CANTON 5248 Dressler Rd NW Mellet Plaza 3826 W Tuscarawas
CENTERVILLE 2026 Miamisburg-Centerville Rd
CINCINNATI 9725 Montgomery, 16-18 Conventon Way (on Skywalk)
CLEVELAND 419 Euclid (Downtwn) 27561 Euclid Ave
COLUMBUS 862 S Hamilton, Great Eastern S/C, The Patio Shop Ctr 4661 Karl Rd 400 N High St
DAYTON Northwest Plaza 3279 West Siebenthaler
ELYNIA 286 Midway Blvd
FAIRFIELD 7255 Dixie Hwy (Ve Mi North of I-275)
NORTH OLMSTED Great Northern S/C
PARMA 7561 W Ridgewood Dr
TOLEDO 5844 W Central Ave , Brownstone Plaza, 1724 S Reynolds Rd
YOUNGSTOWN Union Square Plaza 2543 Belmont Ave

OKLAHOMA
OKLAHOMA CITY 4732 SE 29th St Springdale S/C, 4469 NW 50th 1101 SW 59th St
TULSA 7218 & 7220 E 41st St

OREGON
EUGENE 390 Coburg Rd
PORTLAND 7463 SW Barbur Blvd 9131 SE Powell 3rd and Washington Sts (Downtown)
SALEM Salem Plaza 403 Center

PENNSYLVANIA
ALLENTOWN Crest Plaza S/C Cedar Crest Blvd US 22
BALA CYNWYD 67 E City Line Ave
EASTON 25th St Shopping Center
ELKINS PARK Elkins Park Square, 8080 Old York Rd
ERIE 5755 Peach St
HARRISSURG Union Deposit Mall, Union Deposit Rd #17
LANCASTER Park City Plaza US 30
MONROEVILLE 3826 Wm Penn Hwy
MONTGOMERYVILLE Airport Sq , Rt 309
PHILADELPHIA 7542 Castor Ave., 1002 Chestnut St 1801 Market St., 18 Penn Center
PITTSBURGH 5775 Baptist Rd , Hills Plaza 303 Smithfield St , 4643 Baum Blvd , 4768 McKnight Rd
SCRANTON 206 Meadow Ave
WILKES BARRE, 23 W Market St
WYOMISSING Berkshire Mall West, 1101 Woodland Rd
YORK York County Shopping Center

PUERTO RICO
HATO REY 243 Franklin D Roosevelt Ave

RHODE ISLAND
E. PROVIDENCE 850 Waterman Ave
PROVIDENCE 177 Union St.

SOUTH CAROLINA
COLUMBIA Old Sears Bldg 1001 Harden St
GREENVILLE N Hills S/C
N CHARLESTON 5900 Rivers Ave

SOUTH DAKOTA
SIOUX FALLS 1700 S Minnesota at 25th

TENNESSEE
CHATTANOOGA 636 Northgate Mall
JOHNSON CITY Peerless Center
KNOXVILLE Cedar Bluff S/C 9123 Executive Park Dr
MEMPHIS 4665 American Way, 1997 Union Ave
NASHVILLE 2145 Franklin Pike Rivergate Plaza

TEXAS
AMARILLO Wellington Sq S/C, 1619 S Kentucky
ARLINGTON 2500 E Randol Mill, Suite 113
AUSTIN 8784 E Research Blvd Southwood Mall, 1501 Ben White Blvd
BROWNSVILLE 1639 Price Rd (Hwy 77)
BEAUMONT 5330 Eastex Fwy
COLLEGE STATION 2414 Texas Ave South
CORPUS CHRISTI 1711 S Staple St
DALLAS 15340 Dallas Pkwy , Suite 1100 2930 W Northwest Hwy , 1517 Main St., 2588 Royal Ln
EL PASO 9515 Gateway West, Kern Plaza Shopping Center 3100 N Mesa
FT. WORTH 231 One Tandy Center, 2801 Alta Mere
GALVESTON 5924 Broadway
HARLINGEN 1514 S Hwy 77 Sunshine Strip
HOUSTON 211C-FM 1960 10543 Gulf Fwy 5900 North Fwy , 6813 SW Fwy 809 Dallas St Holland Square Center, 10620 East Freeway Champion Forest Plaza Champion Forest Dr and F M 1960 West, 1018 Gessner, 3278 South Loop West (So Main at 610)
HUMBLE 19300 "B" Hwy 59 (at FM 1960)
HURST Northeast Mall
IRVING 2011 West Airport Fwy
LAREDO 102 East Calton Rd
LUBBOCK 3625 34th St
MEDDVITE Town East Crossing Shop Ctr
ODESSA 1613 A East 8th Street
RICHARDSON Fleetwood Sq S/C, 202 W Campbell Rd 320 S Central Expr
SAN ANTONIO 6018 West Ave , 4249 Centergate, Riverbend Parking Garage, 211 W Market St (Downtown)
WICHITA FALLS 1720-A 9th St

UTAH
OGDEN K-Mart Shopping Center, 3672 Wall Ave
ONEM Grand Central Plaza 384 East & 1300 South
MURRAY 6051 S State Ave
SALT LAKE CITY 301 South State St

VIRGINIA
ALEXANDRIA 3425 King St at Quaker Ln
ARLINGTON Crystal City 2301 So Jefferson Davis Hwy
FAIRFAX Westfax Center 11027 Lee Hwy
LYNCHBURG Hill s Plaza, Ward's Rd
NEWPORT NEWS Newmarket South Shop Ctr
NORFOLK 5731 Poplar Hall Dr Wards Corner 122 E Little Creek Rd
RICHMOND Willow Lawn S/C, 1617 Willow Lawn Dr 7725 Midlothian Turnpike
ROANOKE Franklin Bldg 3561 Franklin Rd S W
ROSSLYN 1911 N Ft Myer Dr at Rt 29

WASHINGTON
BELLEVUE Crossroads Mall North East 8th & 156 St
BELLINGHAM 1111 Cornwall Ave Suite B & C
FEDERAL WAY 33505 Pacific Hwy South
OLYMPIA 108 N Wilson
SEATTLE 18405 Aurora Ave N 1521 3rd Ave , 503 Way NE
SPOKANE 7702 N Division, E 12412 Sprague
TACOMA 7030 S Sprague
TUKWILA 15425 53rd Ave S
YAKIMA 1111 N First St

WEST VIRGINIA
DUNBAR, Dunbar Village Shop Ctr
HUNTINGTON 2701½ 5th Ave

WISCONSIN
APPLETON 2310 West College Ave
MADISON 57 West Towne Mall
MILWAUKEE 6450 N 76th St , 729 N Milwaukee (Downtown)
WEST ALLIS 2717 South 108th St

Radio Shack

TRS-80 Microcomputer News
P.O. Box 2910
Fort Wo , ; 76113-2910

BULK RATE
U.S. POSTAGE
PAID
Radio Shack
A Div. of Tandy Corp.

ADDRESS CHANGE
☐ **Remove from list** ☐ **Change as shown**
Please detach address label and mail to address shown above